Scientific Computing, Modeling & Simulation
Savitribai Phule Pune University

Master of Technology (M.Tech.)
Programme in Modeling and Simulation

Machine Learning Mini Project Report

Hybrid Movie Recommender System Combining Content-Based and Collaborative Filtering Methods

Sarang Pekhale
MT2212

Academic Year 2022-24

Scientific Computing, Modeling & Simulation
Savitribai Phule Pune University

Certificate

This is certify that this report titled

Hybrid Movie Recommender System Combining Content-Based and Collaborative Filtering Methods

authored by

Sarang Pekhale (MT2212)

describes the project work carried out by the author under our supervision during the period from Sept 2023 to Nov 2023. This work represents the project component of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Department of Scientific Computing, Modeling & Simulation, Savitribai Phule Pune University.

Mihir Arjunwadkar, Faculty
SCMS-SPPU, Pune, India

Arun Banpurkar, Head
SCMS-SPPU, Pune, India

Author's Declaration

This document titled

Hybrid Movie Recommender System Combining Content-Based and Collaborative Filtering Methods

authored by me is an authentic report of the project work carried out by me as part of the Master of Technology (M.Tech.) Programme in Modeling and Simulation at the Department of Scientific Computing, Modeling & Simulation, Savitribai Phule Pune University. In writing this report, I have taken reasonable and adequate care to ensure that material borrowed from sources such as books, research papers, internet, etc., is acknowledged as per accepted academic norms and practices in this regard. I have read and understood the University's policy on plagiarism (`http://unipune.ac.in/administration_files/pdf/Plagiarism_Policy_University_14-5-12.pdf`).

Sarang Pekhale
MT2212

Abstract

This project presents a hybrid movie recommender system that combines content-based and collaborative filtering methods. By fusing movie metadata with user interactions, the system enhances personalized movie recommendations. The hybrid model outperforms standalone approaches in accuracy and diversity. This project contributes to the field of recommendation systems, showcasing the value of hybrid models in improving the user experience. Future work may involve further optimization and real-world deployment.

Acknowledgements

I am indebted to **Prof. Dr. Mihir Arjunwadkar** for their guidance on a daily basis during this project, his expertise were invaluable in shaping the direction of this work. I would like to express my sincere gratitude to all those who contributed to the successful completion of this project. I also extend my appreciation to my peers and colleagues for their support and collaboration throughout the project. Finally, I am thankful to my family and friends for their unwavering encouragement and understanding during this endeavor.

Contents

Chapter 1

Introduction

1.1 Problem Statement

The challenge is to develop a hybrid movie recommender system that combines content-based and collaborative filtering methods to address the following key issues:

1. **Information Overload**: With the abundance of movies available on streaming platforms, users often struggle to discover movies that cater to their specific tastes. The problem involves sifting through a vast selection of movies to find those that are truly appealing.
2. **Diverse User Preferences**: User preferences for movies are diverse and continually evolving. Traditional recommendation methods may not adequately capture the dynamic nature of user interests.
3. **Cold Start Problem**: For new users who have limited interaction history, it can be challenging to provide meaningful recommendations. The "cold start" problem requires a solution that goes beyond collaborative filtering.
4. **Limited Exploration**: Recommendation systems that solely rely on one technique, such as collaborative filtering or content-based filtering, can lack the ability to provide diverse movie recommendations, often leading to user fatigue and missed opportunities for user engagement.

I will conduct an Exploratory Data Analysis to uncover valuable insights that will help me answer intricate and complex questions such as:

1. Which movies are the most popular based on metrics like vote count and average rating?
2. Are there any trends or patterns in movie popularity over time?
3. How active are users in terms of movie ratings and interactions?
4. Are there user segments with distinct movie preferences or behavior?
5. What are the most common movie genres in the dataset?
6. How do genre preferences vary among different user groups?
7. Are there specific keywords or phrases that are associated with highly rated movies?
8. How has movie production and popularity evolved over time?
9. Are there trends in movie popularity based on release dates, such as seasons or years?

ETC.

1.2 Overview of Movie Recommender Systems

In today's era of digital entertainment, personalized recommendations play a pivotal role in enhancing user experiences. The ability to recommend relevant and engaging content not only keeps users entertained but also contributes significantly to customer satisfaction and retention. Recommender systems have become a cornerstone of various online platforms, assisting users in navigating vast libraries of content with ease. In this context, the development of effective movie recommender systems stands as a compelling challenge and opportunity.

This project embarks on the creation of a hybrid movie recommender system, a culmination of content-based and collaborative filtering methods. The primary objective is to provide movie enthusiasts with recommendations that are not only tailored to their individual preferences but also diverse and engaging. While content-based systems leverage movie metadata such as genre, keywords, and descriptions, collaborative filtering models draw insights from user behavior and preferences. A hybrid approach, as presented in this project, combines the advantages of these two recommendation techniques, allowing for the fine-tuning of accuracy and diversity.

Moreover, this project aims to contribute to the evolving landscape of recommendation systems, emphasizing the practical application of hybrid models in the competitive field of movie recommendations. The lessons learned from this project may serve as a stepping stone for future enhancements, real-world deployment, and the inclusion of user feedback mechanisms for continuous system improvement.

In essence, this project delves into the development of a hybrid movie recommender system, showcasing the potential of combining various recommendation techniques to offer more precise and diversified movie suggestions. It addresses the dynamic and competitive landscape of digital entertainment while prioritizing the user's individual viewing preferences.

1.2.1 Content-Based Method

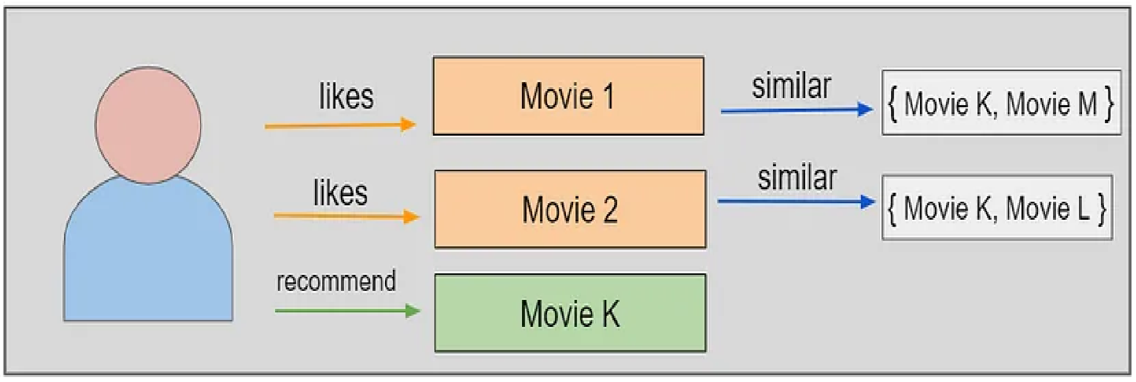


Figure 1.1: Content Based Movie Recommender System

In the ever-expanding landscape of digital entertainment, the demand for personalized and relevant content recommendations has never been more apparent. Content-based movie recommender systems have emerged as a solution to this need, offering users movie suggestions based on their individual tastes and preferences. Unlike collaborative filtering methods that rely on user interactions, content-based systems focus on the intrinsic characteristics of movies, such as genre, keywords, and descriptions. This approach opens up new avenues for improving user experiences by harnessing the unique features of movies to make tailored recommendations.

How Content-Based Movie Recommender Systems Work

Content-based movie recommender systems operate on the principle of analyzing the content features associated with each movie in the catalog. The process can be broken down into the following steps:

1. **Feature Extraction**:Relevant movie features are extracted from available metadata, including genre information, keywords, and textual descriptions. These features serve as the basis for understanding the content of each movie.
2. **User Profile Creation**:The system builds a user profile by analyzing the user's historical movie preferences, ratings, or interactions. This user profile encapsulates the types of movies the user has previously shown interest in.
3. **Content Matching**: The system compares the content features of movies in the catalog with the user profile. It identifies movies that share similar content attributes to those favored by the user.
4. **Recommendation Generation**: Based on the content matching process, the system generates a list of movie recommendations. These recommendations are tailored to the user's content preferences and can encompass a range of movies that align with their interests.

Implementation of Content-Based Movie Recommender Systems

The successful implementation of a content-based movie recommender system involves several key elements:

1. **Data Collection and Preprocessing**:Gather movie data, including metadata like genre, keywords, and descriptions. Clean and preprocess the data to ensure consistency and quality.
2. **Feature Engineering**: Extract relevant features from the movie data, including text analysis for keywords and descriptions.
3. **User Profiling**:Create user profiles that represent their content preferences based on past interactions with movies.
4. **Content Matching Algorithm**:Implement content matching algorithms that determine the similarity between user profiles and movie content features.
5. **Recommendation Generation**:Use the matching results to generate personalized movie recommendations for users.

Applications of Content-Based Movie Recommender Systems

AContent-based movie recommender systems find applications in various domains, including:

1. **Movie Streaming Platforms**:These systems enhance user engagement by offering movie recommendations aligned with individual tastes, thereby increasing user retention.

2. **E-commerce Platforms**:Content-based recommendations are not limited to movies and extend to products. E-commerce platforms use similar techniques to recommend products based on user preferences.

3. **Information Retrieval**:Content-based filtering is applied in information retrieval systems to recommend relevant articles, news, or content to readers.

4. **Advertising and Marketing**:Content-based recommendations are utilized in ad targeting and personalized marketing campaigns to present users with content that matches their interests.

In summary, content-based movie recommender systems use movie features to create personalized recommendations. These systems have broad applications and significantly impact user engagement, satisfaction, and business success in various domains.

1.2.2 Collaborative Filtering Method

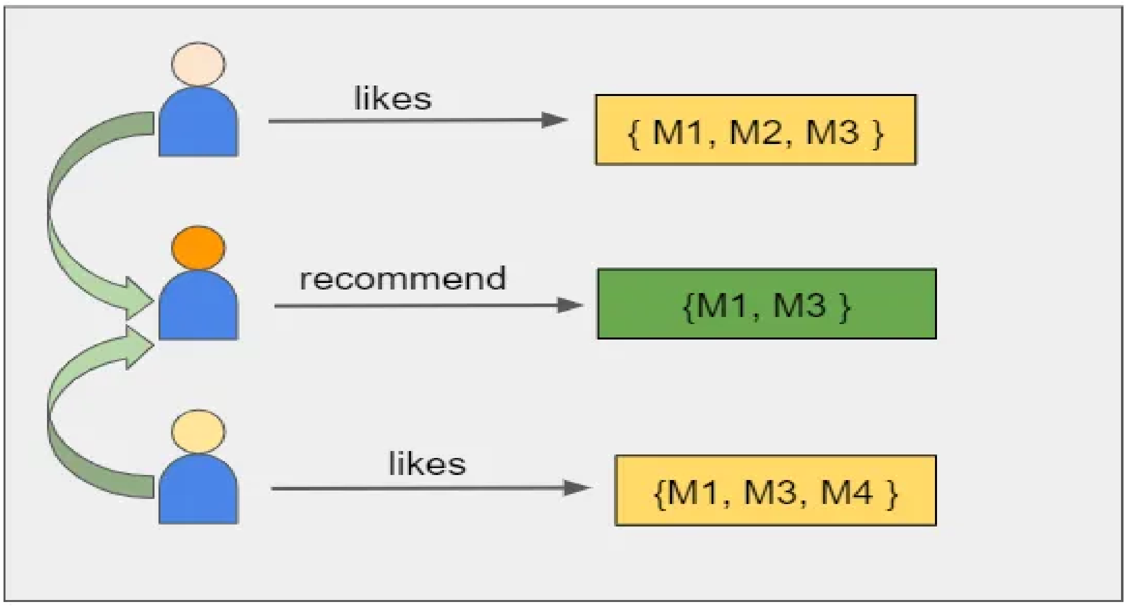


Figure 1.2: Content Based Movie Recommender System

In the era of digital entertainment, the need for personalized and engaging movie recommendations is ever-present. Collaborative filtering-based movie recommender systems have emerged as a solution to this need, focusing on the power of collective user behavior and preferences to offer tailored movie suggestions. Unlike content-based methods that rely on movie features, collaborative filtering systems analyze user interactions and similarity to connect users with movies they are likely to enjoy. This approach holds the promise of enhancing the user experience through recommendations derived from shared preferences and behaviors.

How Collaborative Filtering-Based Movie Recommender Systems Work

Collaborative filtering-based movie recommender systems operate by understanding the underlying patterns in user interactions. The process can be summarized as follows:

1. **User-Item Interaction Matrix**:The system creates a matrix where rows represent users and columns represent movies. The entries in this matrix denote user interactions, such as ratings or views, and reveal user preferences.

2. **User-Based Collaborative Filtering**:This method identifies users who share similar preferences with the target user. It calculates user similarity scores and recommends movies that like-minded users have enjoyed.

3. **Item-Based Collaborative Filtering**:This approach identifies movies that are similar to those the user has previously interacted with based on patterns in the interaction matrix. It recommends movies based on the similarity of item interactions.

4. **Recommendation Generation**:The system generates movie recommendations based on the collaborative filtering technique chosen (user-based or item-based). These recommendations align with user preferences as inferred from the user interaction matrix.

Implementation of Collaborative Filtering-Based Movie Recommender Systems

Implementing a collaborative filtering-based movie recommender system involves the following key components:

1. **Data Collection**:Gather user interaction data, which may include ratings, views, or other forms of feedback.

2. **User-Item Interaction Matrix**:Create a matrix that represents the user-movie interactions, with rows representing users and columns representing movies.

3. **Item Similarity Calculation**:If using item-based collaborative filtering, calculate item similarities based on user interaction patterns.

4. **Recommendation Generation**:Utilize the calculated similarities to generate personalized movie recommendations for users.

Applications of Collaborative Filtering-Based Movie Recommender Systems

Collaborative filtering-based movie recommender systems find applications in various domains, including:

1. **Movie Streaming Platforms**:These systems enhance user engagement by providing movie recommendations based on collective user behavior, thereby increasing user satisfaction and retention.

2. **E-commerce Platforms**:Collaborative filtering is used to recommend products based on the preferences of other users who have shown similar shopping behavior.

3. **Social Media**:It is applied to suggest new connections, content, or groups to users on social media platforms.

4. **News and Article Recommendations**:News aggregators and content platforms use collaborative filtering to suggest articles and news items based on user interests.

In summary, collaborative filtering-based movie recommender systems analyze user interactions to provide personalized recommendations. These systems have broad applications and significantly impact user engagement, satisfaction, and business success in various domains.

1.3 Data : TMDB 5000 Movie Dataset

1.3.1 Source of the TMDB 5000 Movie Dataset

The TMDB 5000 Movie Dataset is a publicly available dataset found on Kaggle, a renowned platform for data science and machine learning datasets. It originates from The Movie Database (TMDb), a comprehensive online resource dedicated to movies, television programs, and the entertainment industry. This dataset has been thoughtfully compiled and made accessible on Kaggle for data science enthusiasts and researchers interested in exploring and analyzing movie-related data.

1.3.2 Description of the TMDB 5000 Movie Dataset

The TMDB 5000 Movie Dataset is a treasure trove of information about a diverse array of movies. It provides extensive details about various films, making it an invaluable resource for projects and analyses related to the world of cinema. The dataset typically includes the following key features and attributes:

1. **Movie Metadata**:The dataset encompasses essential information about each movie, including its title, release date, and genre. This metadata serves as the foundational description of each film.

2. **Crew and Cast Information**:Detailed data on the individuals behind the scenes and in front of the camera are included. This covers directors, producers, writers, actors, and other crew members who contributed to the creation of the movies. This wealth of information is instrumental in exploring the influence of cast and crew on movie recommendations.

3. **Keywords**:The dataset contains a rich repository of keywords associated with each movie. These keywords provide valuable insights into the themes, topics, and genres addressed in the films. Keyword information can play a crucial role in content-based recommendations, aligning movie themes with user preferences.

4. **User Ratings and Popularity Metrics**:Each movie's user ratings, popularity metrics, and revenue details are available. This enables the evaluation of movie popularity and user preferences. Ratings data can be utilized in collaborative filtering-based recommendations, allowing for the assessment of movie preferences shared among users.

5. **Production Companies**:The dataset features details about the production companies responsible for each movie. Understanding the production company behind a film can offer insights into industry dynamics and user preferences.

6. **Runtime and Budget Information**:Specifics about the duration of each movie and its production budget are included. These details can be pertinent in guiding users toward movies that align with their preferences for film length and production scale.

7. **Overview and Tagline**: Descriptive overviews and taglines for movies provide concise summaries and thematic highlights that are valuable in understanding and recommending movies.

8. **Spoken Languages and Release Regions**:Information about the languages spoken in the movies and their release regions contributes to a more refined recommendation process that aligns with users' language and regional preferences.

1.3.3 Relevance to Movie Recommender System

The TMDB 5000 Movie Dataset is exceptionally relevant to movie recommender system project for several compelling reasons:

1. **Rich Movie Feature Data**: The dataset offers a comprehensive array of movie features, including genres, keywords, crew, and cast information. These attributes are fundamental for content-based movie recommendations within your system.
2. **User Ratings and Popularity**:User ratings and popularity metrics permit the evaluation of movie popularity and the potential for collaborative filtering-based recommendations that align with user preferences and trends.
3. **Cast and Crew Details**:Information about the cast and crew provides a powerful tool for enhancing personalization in recommendations. Users with specific preferences for actors, directors, or producers can receive tailored movie suggestions.
4. **Diverse Movie Selection**:With an extensive collection of movies, the dataset enables a wide spectrum of recommendations, catering to a diverse array of user tastes and preferences.
5. **Real-World Relevance**: Leveraging data from TMDB, a renowned movie database, imparts a real-world dimension to your project. The dataset mirrors the type of data encountered in production movie recommender systems, offering a solid foundation for building and evaluating your recommender system.
6. **Multilingual and Global Context**:The inclusion of information about spoken languages and release regions allows for the customization of recommendations based on user language preferences and global release patterns, enhancing the user experience.

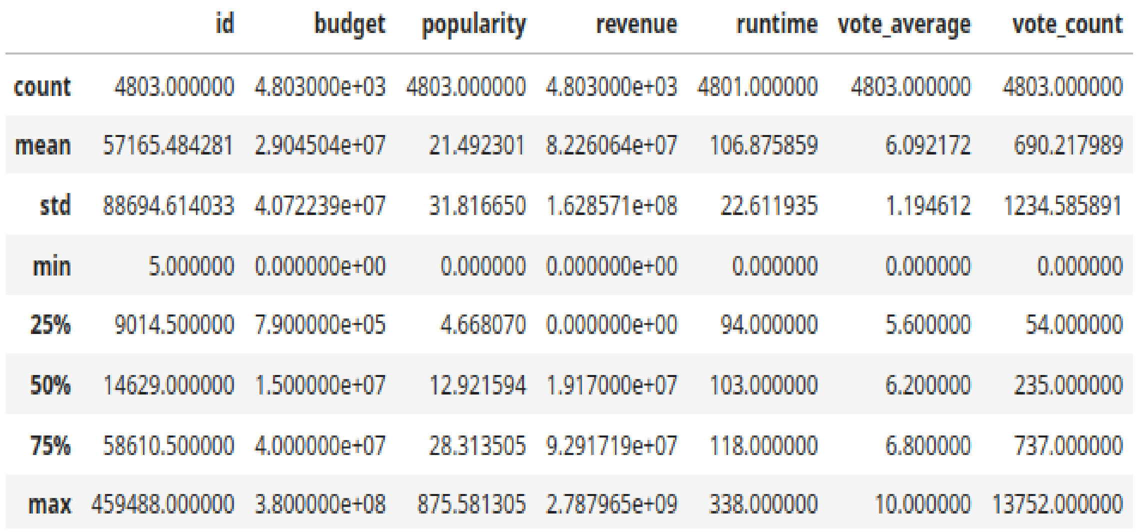
Incorporating the TMDB 5000 Movie Dataset into project equips with a robust and dynamic dataset that enhances the accuracy, quality, and personalization of movie recommendations offered to users. The dataset's versatility and depth of information align with the goals of a comprehensive movie recommender system.

1.3.4 Data Preprocessing for EDA

In this section, I will describe the process of importing, cleaning, and preprocessing the data necessary for our analysis. This is a critical step in any data analysis project as it ensures that the data is in a usable and structured format for subsequent tasks.

- **Libraries Required**:
- We utilized several Python libraries to facilitate this data handling process:
    1. **numpy**: We used the NumPy library for working with arrays, matrices, and performing various mathematical functions. This library is invaluable for efficient numerical operations.
    2. **pandas**: We leveraged the pandas library, which provides data structures such as DataFrames and Series. DataFrames are particularly useful for efficiently handling structured data, such as CSV files, and performing data manipulation tasks.
    3. **ast**: We used the ast library, which stands for "Abstract Syntax Trees." This library is helpful for parsing and evaluating literal expressions within the data.

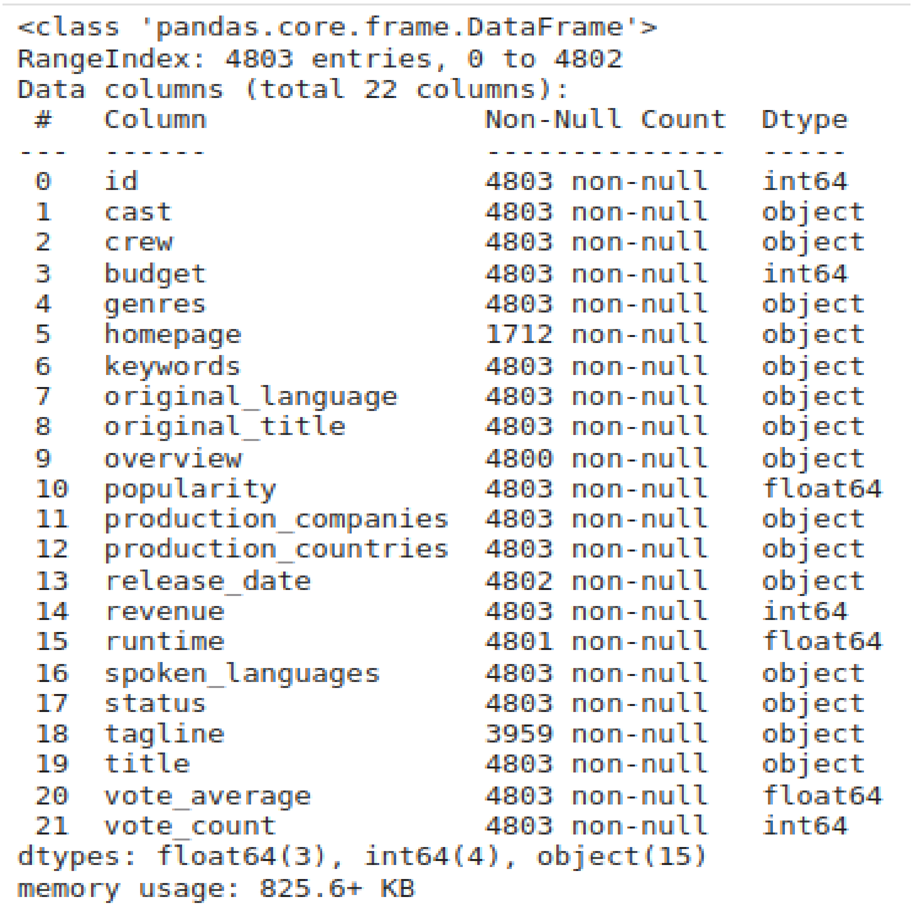
- **Importing Data**: We utilized the pandas library in Python to import two CSV files into our environment. Subsequently, we established two distinct DataFrames, namely **'df-credits'** and **'df-movies'** which were employed to store and manage the data contained within these files. These DataFrames served as the foundation for our data analysis efforts.
- **Data size or dimensions**: The data size or dimensions of our datasets are as follows:
  1. **df-credits**: For 'df-credits,' we have a dataset with dimensions of (4803 rows, 4 columns).
  2. **df-movies**: In the case of 'df-movies,' the dataset dimensions are (4803 rows, 20 columns).
- This information provides an overview of the size and structure of our data, which is essential for understanding the scope of our analysis.
- In the first DataFrame, we renamed the 'movie-id' column to 'id' for consistency.
- To avoid duplicate columns, we removed the 'title' column from the CREDITS dataset.
- Subsequently, we performed a merge operation on the two DataFrames, CREDITS and MOVIES, using the common 'id' column.
- The merged DataFrame, referred to as 'df,' now has dimensions of (4803 rows, 22 columns). These actions streamlined our data preparation and consolidation process, resulting in a merged dataset ready for further analysis.
- A table of descriptive statistics of numerical columns of the TMDB 5000 dataset. The table shows the count, mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, and maximum values for the id, budget, popularity, revenue, runtime, vote average, and vote count columns.



	id	budget	popularity	revenue	runtime	vote_average	vote_count
count	4803.000000	4.803000e+03	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000000
mean	57165.484281	2.904504e+07	21.492301	8.226064e+07	106.875859	6.092172	690.217989
std	88694.614033	4.072239e+07	31.816650	1.628571e+08	22.611935	1.194612	1234.585891
min	5.000000	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	9014.500000	7.900000e+05	4.668070	0.000000e+00	94.000000	5.600000	54.000000
50%	14629.000000	1.500000e+07	12.921594	1.917000e+07	103.000000	6.200000	235.000000
75%	58610.500000	4.000000e+07	28.313505	9.291719e+07	118.000000	6.800000	737.000000
max	459488.000000	3.800000e+08	875.581305	2.787965e+09	338.000000	10.000000	13752.000000

Figure 1.3: Descriptive statistics of Data

- Summary of the DataFrame's basic information. Like Names of the columns in the dataset, Count of non-null values in each column, and Datatype of each column.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    4803 non-null   int64  
 1   cast                  4803 non-null   object 
 2   crew                  4803 non-null   object 
 3   budget                4803 non-null   int64  
 4   genres                4803 non-null   object 
 5   homepage              1712 non-null   object 
 6   keywords              4803 non-null   object 
 7   original_language     4803 non-null   object 
 8   original_title        4803 non-null   object 
 9   overview              4800 non-null   object 
 10  popularity            4803 non-null   float64
 11  production_companies  4803 non-null   object 
 12  production_countries  4803 non-null   object 
 13  release_date          4802 non-null   object 
 14  revenue               4803 non-null   int64  
 15  runtime               4801 non-null   float64
 16  spoken_languages      4803 non-null   object 
 17  status                4803 non-null   object 
 18  tagline               3959 non-null   object 
 19  title                 4803 non-null   object 
 20  vote_average          4803 non-null   float64
 21  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(15)
memory usage: 825.6+ KB
```

Figure 1.4: Data Information

- We began by identifying the columns in the DataFrame 'df' that possess numeric data types.
- Our subsequent step involved counting the number of missing (null) values within each column of the DataFrame 'df.'
- For data cleanliness and considering our focus on Hollywood movies where English is the primary language, we removed the 'homepage,' 'tagline,' and 'spoken-languages' columns due to significant null values.
- To further enhance data quality, we eliminated rows with missing values from the 'df' DataFrame.
- We conducted an assessment for both duplicate columns and duplicate rows within the 'df' DataFrame.
- Recognizing that 'original-title' and 'title' are nearly identical, we opted to remove the 'original-title' column.
- The columns present in the 'df' DataFrame are as follows: ['id', 'cast', 'crew', 'budget', 'genres', 'keywords', 'original-language', 'overview', 'popularity', 'production-companies', 'production-countries', 'release-date', 'revenue', 'run-time', 'status', 'title', 'vote-average', 'vote-count'].

- We specifically desired to extract the textual data from the dictionaries in the observations of these columns and proceeded to do so.
- Within the 'df' DataFrame, we focused on the columns: "keywords," "genres," "production-companies," and "production-countries."
- We implemented a function designed to transform a string containing a list of dictionaries into a list of values extracted from a specific key within those dictionaries.
- The ast.literal-eval(obj) function was employed for safe evaluation, allowing us to parse and execute single expressions in a string format containing literal Python data structures.
- We standardized the format of the "release-date" column to "yyyy/mm/dd" to ensure consistency.
- Furthermore, we extracted the individual components of the date, namely "year," "month," and "day," from the "release-date" feature.
- To maintain data integrity and uniqueness, we initially segregated data based on the "title" and assessed if any movie titles were repeated and, if so, how many times.
- We then integrated the movie titles with their release dates to ensure the uniqueness of each observation.
- We converted the JSON-formatted "cast" and "crew" data into a list format. From these features, we selected specific factors, including 'name,' 'gender,' 'job,' and 'department,' for further exploratory data analysis (EDA) purposes.
- Given that a movie can be associated with multiple genres, we split the dataset row-wise based on the "genres" to ensure accurate genre information.
- With the data now cleaned and structured, we are well-prepared to commence our Exploratory Data Analysis (EDA) to gain valuable insights and uncover patterns in the dataset.

Chapter 2

Exploratory Data Analysis

2.1 Exploratory Data Analysis

EDA is our compass as we explore the TMDB 5000 dataset, unraveling user preferences and cinematic details. In this project, it's the foundational step for our hybrid movie recommender system. EDA reveals patterns in user ratings, movie attributes, and interactions, enhancing our system's accuracy and personalization. We aim to craft a movie recommendation experience that seamlessly combines content-based and collaborative filtering, elevating the user's cinematic journey.

1. **Pairplots**: a grid of scatterplots for numeric columns Index(['id', 'budget', 'popularity', 'revenue', 'runtime', 'vote-average', 'vote-count'], helping to visualize potential associations and patterns between these attributes. We can see that there are several variables that have a strong positive correlation with each other, such as budget and revenue, and popularity and revenue. There are also some variables that have a weak or no correlation with each other, such as vote average and revenue

    - Here we can see patterns in the plot of 'budget' vs 'popularity', 'revenue' and 'vote-count'.
    - There is some pattern in the plot of 'popularity', vs 'revenue' and 'vote-count'.
    - 'revenue' vs 'vote-count' plot also have noticeable pattern init.

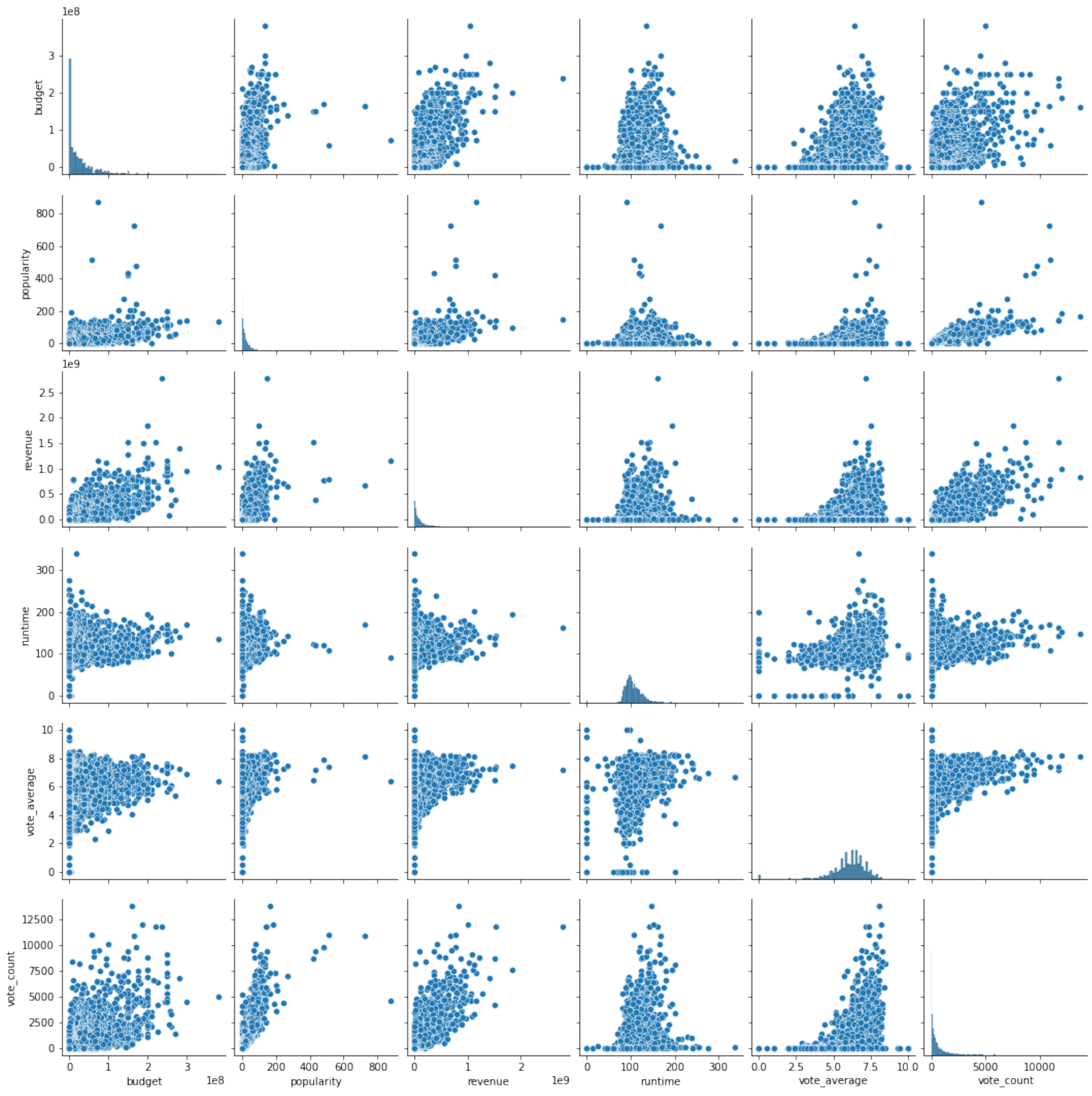


Figure 2.1: Pairplot of Data

- To enhance the plots' variation, i will form pairplots again but now taking log scale on both sides.

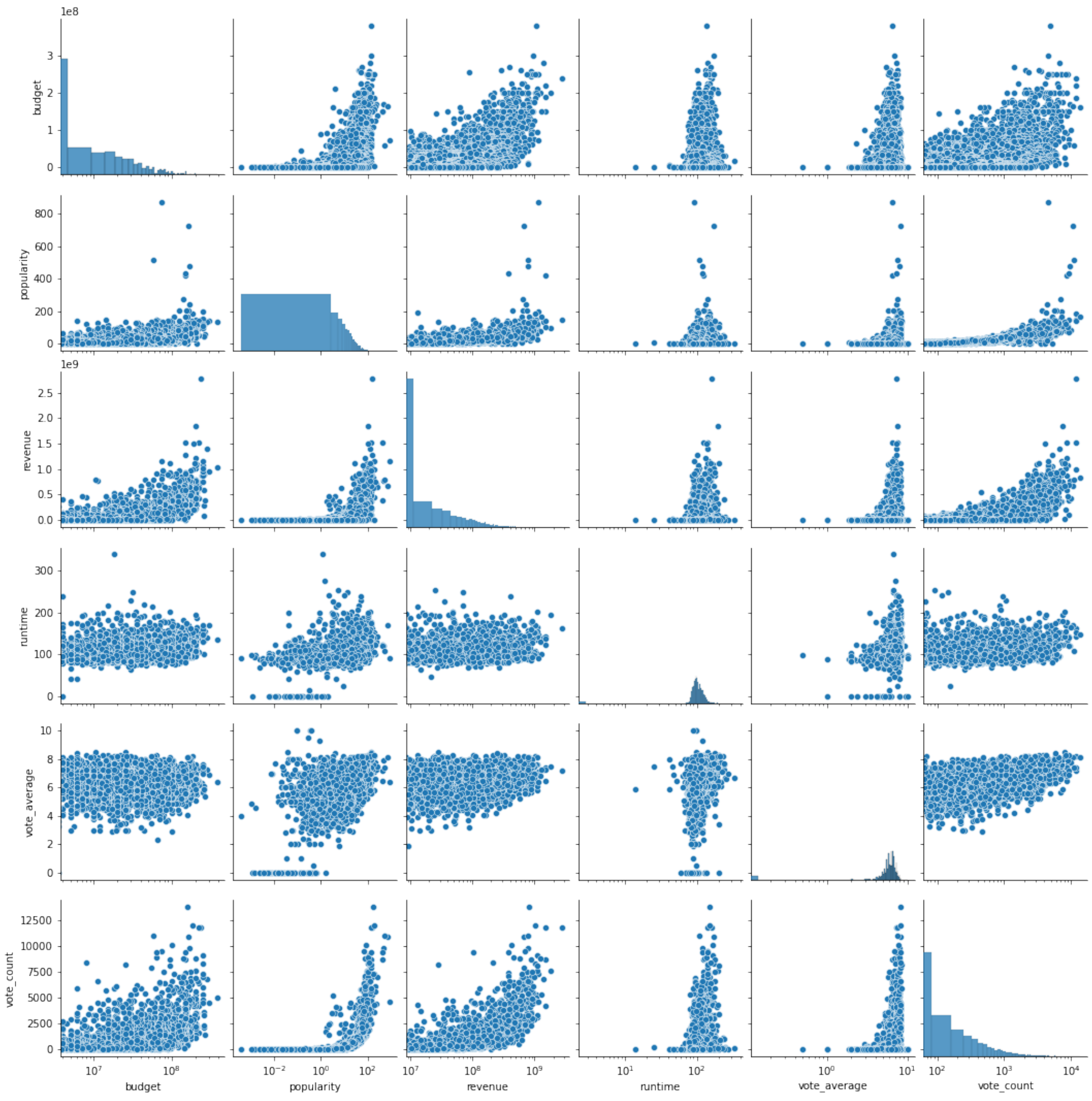


Figure 2.2: Pairplot of Log scaled Data

- **Correlation matrix as a heatmap**: A correlation heatmap of the tmdb 5000 movie dataset. The heatmap shows the correlation between different numerical variables such as budget, popularity, revenue, runtime, vote average and vote count. We can see that there is a strong positive correlation between budget and revenue. There is also a moderate positive correlation between popularity and revenue. However, there is no significant correlation between vote average and revenue.

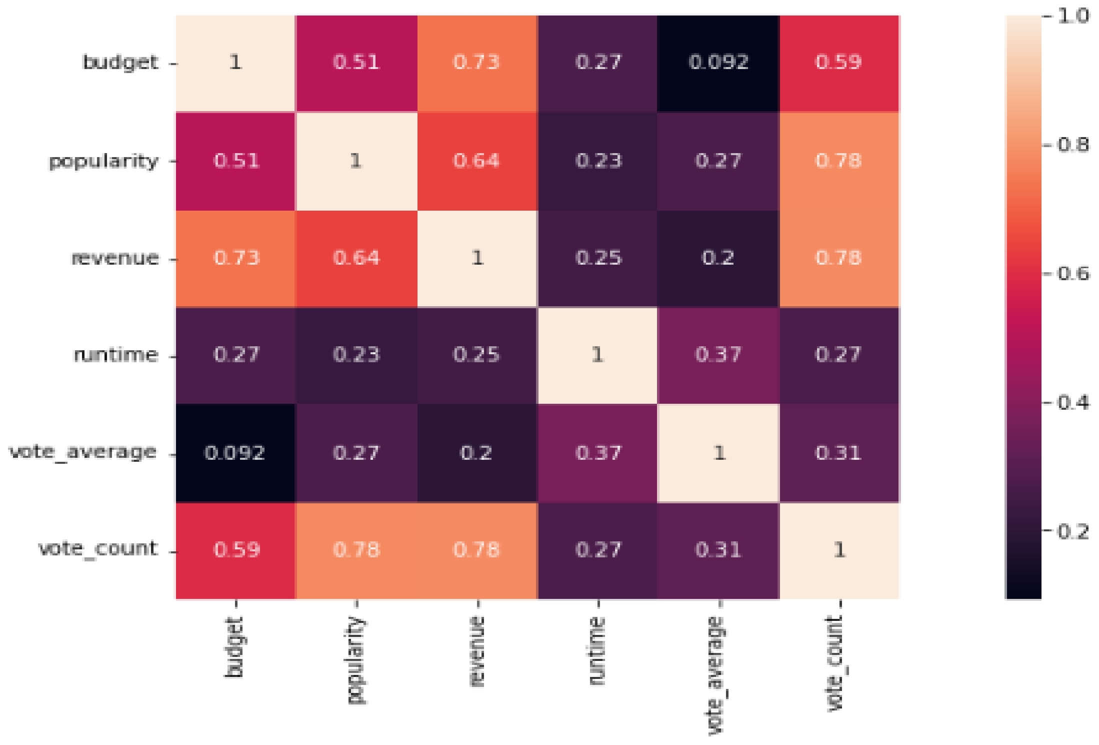


Figure 2.3: Correlation Matrix of Data

2. **Movies Runtime**: A histogram and a boxplot of movie runtime. The histogram shows the distribution of movie runtime, while the boxplot shows the median, quartiles, and outliers of the distribution. We can see that most movies have a runtime between 100 and 150 minutes, as indicated by the peak in the histogram. The boxplot shows that the median runtime is around 100 minutes, with the lower quartile around 90 minutes and the upper quartile around 120 minutes. There are also some outliers in the data.

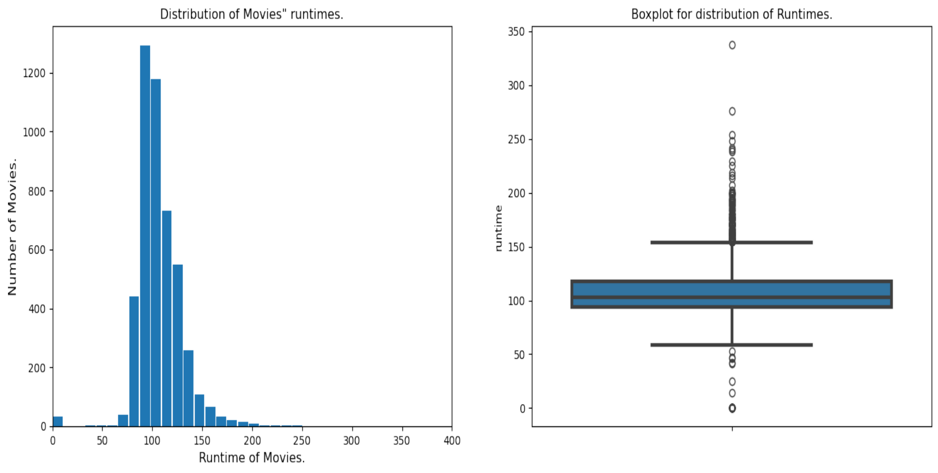


Figure 2.4: Movies Runtime

3. **Budget vs Revenue**: Scatter plot of Revenue and Budget. The x-axis represents the log of movie budget and the y-axis represents the log of movie revenue. There is a trend line that shows a positive correlation between the two variables. In this case, it shows how the revenue of a movie is related to its budget. The positive correlation between the two variables indicates that as the budget of a movie increases, so does its revenue

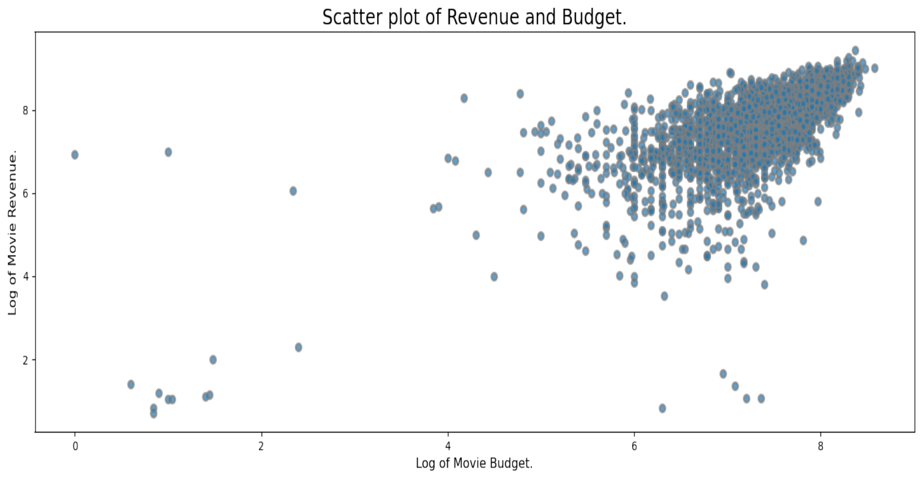


Figure 2.5: Budget vs Revenue

4. **Movies Budget**: Histogram shows the distribution of movie budgets. The x-axis represents the budget in 100 million dollars and the y-axis represents the number of movies. The histogram shows that most movies have a budget of less than 500 million dollars, with a few outliers that have a budget of more than 3 billion dollars.

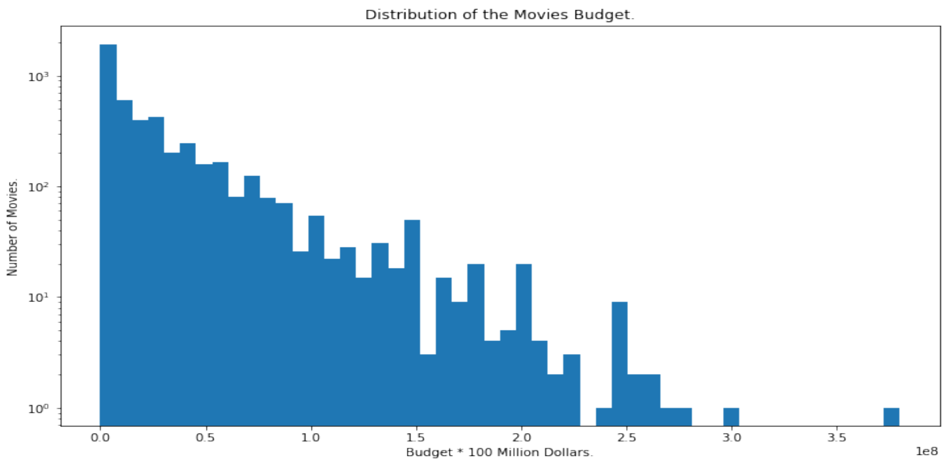


Figure 2.6: Movies Budget

5. **Movies Revenues**: The histogram shows the distribution of movie revenues in billions of dollars. The x-axis represents the revenue in billions of dollars, while the y-axis represents

the number of movies. The bars decreases in height as the revenue increases. The highest bar is at the range of 0.0 billion dollar mark and the lowest bar is at the range of 2.5 billion dollar mark.

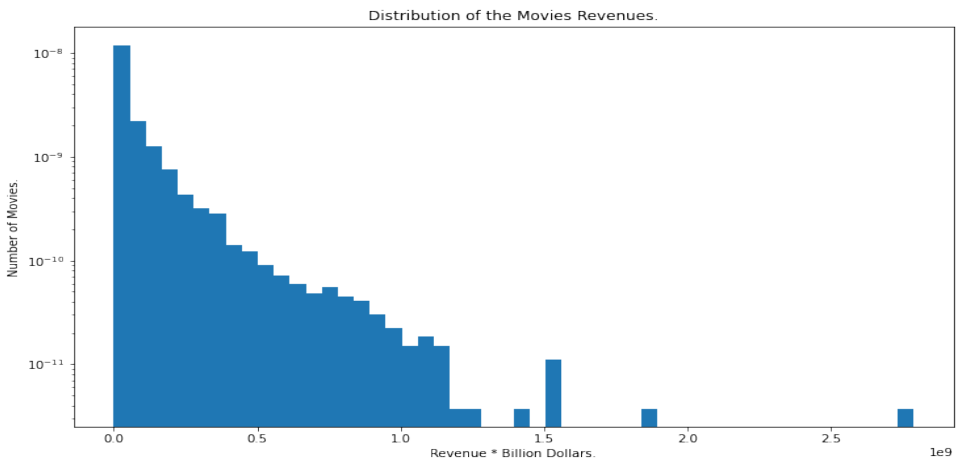


Figure 2.7: Movies Revenues

6. **Movies by Generes**: A pie chart that shows the number of movies per genre in the TMDB 5000 movie dataset. The chart has 18 different genres represented, with the largest genre being Drama, followed by Comedy and Thriller and Action, covering almost 50% share in movies produced combined. The smallest genres are War, History, and Music and more.

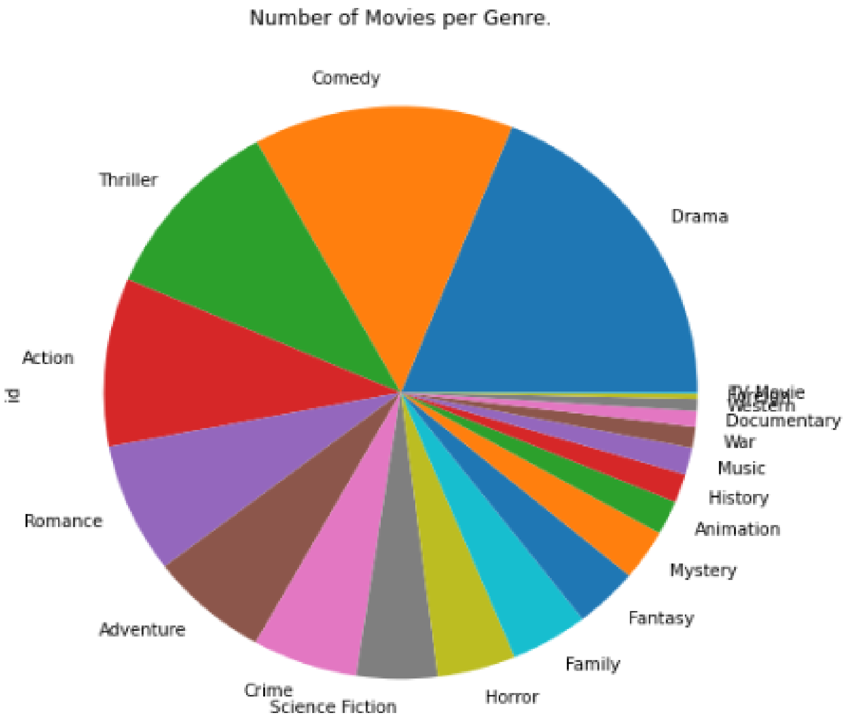


Figure 2.8: Movies by Generes

7. **Movies Revenues by Generes**: Analysing the revenues collected by each genere so far by plotting multiple boxplot. This plot visualizes the distribution of movie revenues (on the y-axis) across different genres (on the x-axis) while filtering out movies with revenue less than zero. The boxen plot provides insights into the revenue distribution within each genre category, helping to identify variations and trends in movie earnings among different genres. The highest revenue is for the Adventure genre, followed by Fantasy and Science Fiction. The lowest revenue is for the Foreign, Documentary, and Western genre.

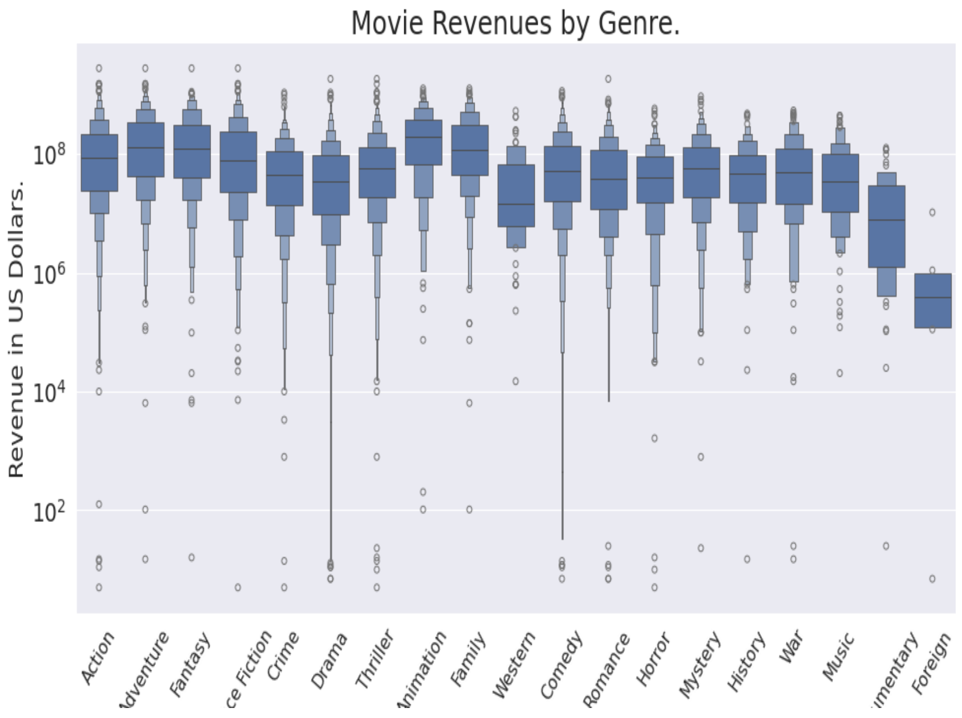


Figure 2.9: Movies Revenues by Generes

8. **Movie Revenue by Genres**: This pie chart is a visual representation of the revenues per movie genre of the TMDB 5000 movies dataset. The chart has 16 different genres represented, with the largest section being for the Adventure genre, which contributes approximately 18% of the total revenue. The smallest section is for the Western genre, which contributes approximately 2% of the total revenue.

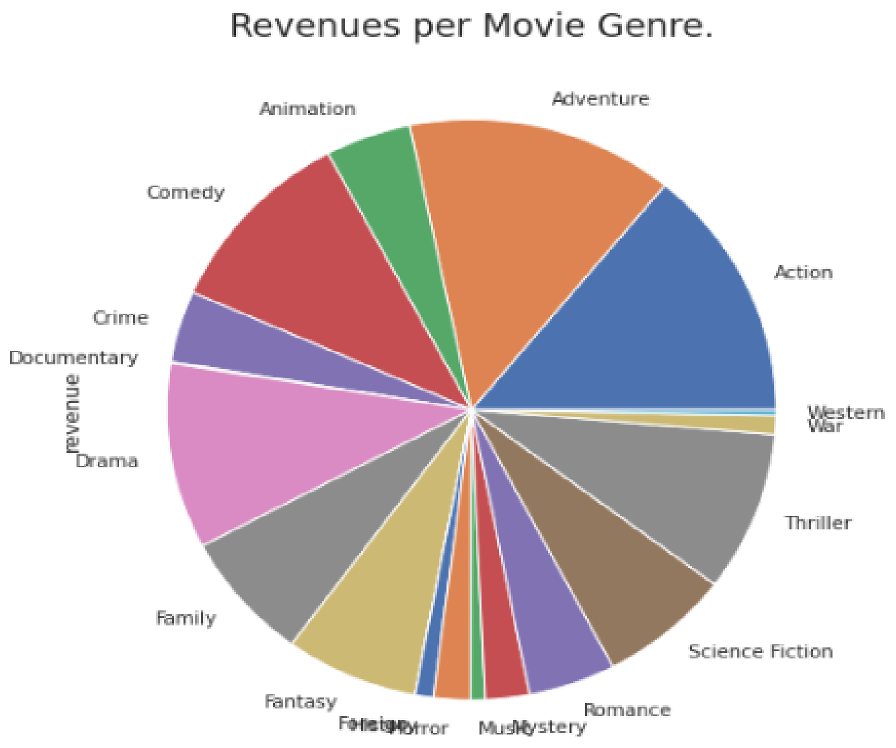


Figure 2.10: Movie Revenue by Genres

9. **Movies Scores by Genres**: It shows the distribution of movie scores for different genres. The genres are on the x-axis and the movie scores are on the y-axis. The boxenplot shows the median, quartiles, and outliers for each genre. The median is the line in the middle of the box, the quartiles are the top and bottom of the box, and the outliers are the points outside the box. The boxenplot shows that the median movie score for all genres is around 6, with the exception of TV Movie, which has a median score of around 5. The boxenplot also shows that the genres with the highest scores are Animation, History, and War, while the genres with the lowest scores are TV Movie and Horror.

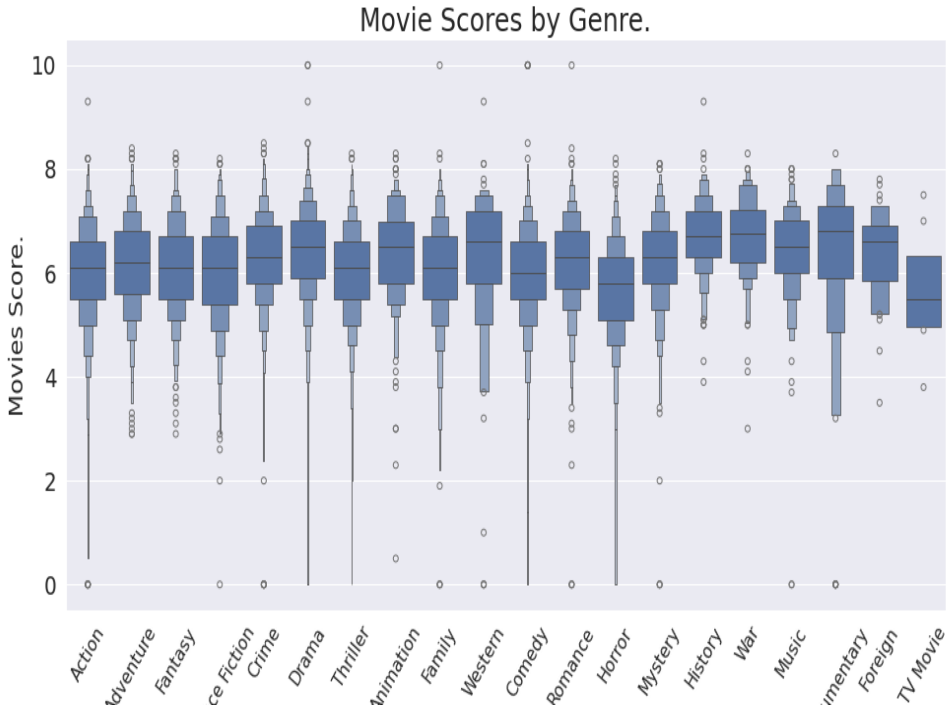


Figure 2.11: Movies Scores by Genres

10. **Movies Popularity and Budget**: This scatterplot shows the relationship between movie popularity and budget for the TMDB 5000 movies dataset. The x-axis represents the budget in millions of dollars, while the y-axis represents the popularity. The majority of the movies have a budget below $100 million and a popularity below 200. However, there are some outliers, such as “Jurassic World” and “Furious 7”, which have a high budget and high popularity. “The Lone Ranger” and “John Carter” have a high budget but low popularity.

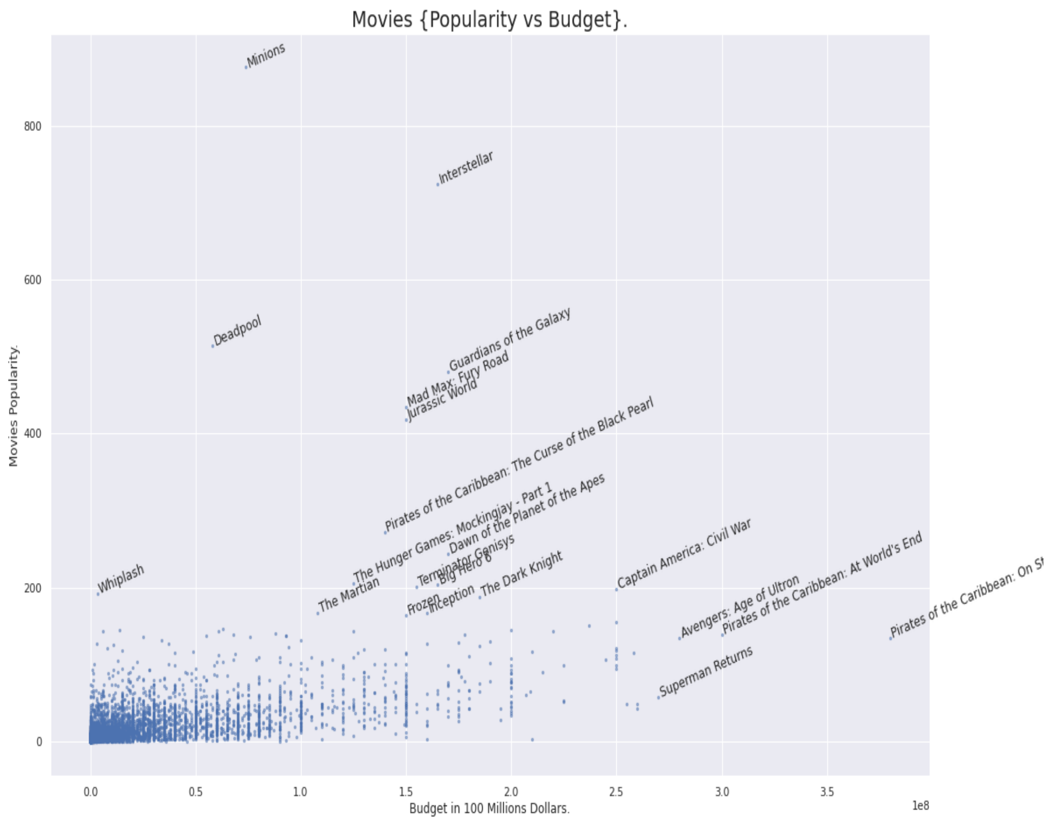


Figure 2.12: Movies Popularity and Budget

11. **Movies Revenue and Release Year**: The x-axis represents time in years, starting from 1920 and ending in 2020. The y-axis represents the movie's revenue in billion dollars, starting from 0 and ending at 2.5. According to the graph, the revenue of movies has increased over time, with a few spikes in certain years. However, it is important to note that this graph only represents the revenue of movies in the TMDB 5000 movies dataset and may not be representative of the entire movie industry.

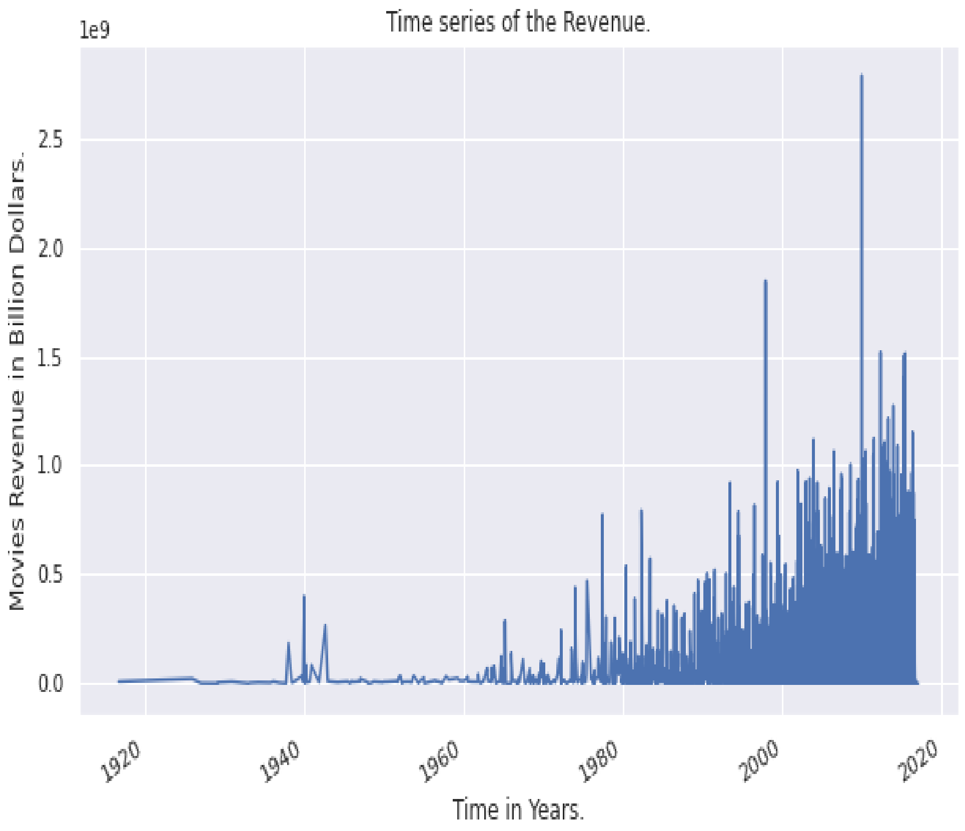


Figure 2.13: Movies Revenue and Release Year

12. **Movies Revenue and Release Year with Genres**: A bar graph that shows the revenue generated by movies per genre and year. The x-axis represents the year of release, and the y-axis represents the movie's revenue in dollars. The bars are color-coded according to the movie's genre. The genres included are Action, Adventure, Animation, Comedy, Crime, Drama, Family, Fantasy, Foreign, History, Horror, Music, Mystery, Romance, Science Fiction, Thriller, War, and Western. The graph shows data from 1990 to 2015. According to the graph, the highest revenue is generated by movies in the Action genre, followed by Adventure and Animation. The lowest revenue is generated by movies in the Foreign genre. The graph also shows that the revenue generated by movies has increased over time, with a few spikes in certain years.

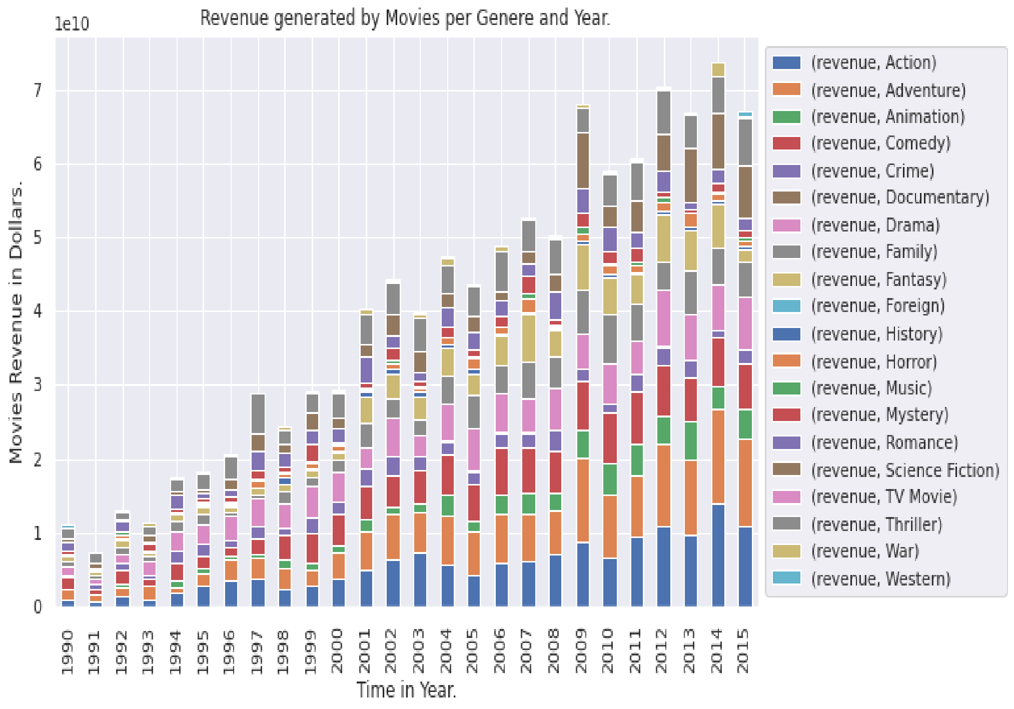


Figure 2.14: Movies Revenue and Release Year with Genres

13. **Movies Revenue and Adjusted Revenue vs Movie Year**: The x-axis represents the time in years, from 1920 to 2020. The y-axis represents the revenue in billions of dollars. The blue line represents the revenue of the top 5000 movies in the TMDB dataset, and the orange line represents the adjusted revenue of the top 5000 movies in the TMDB dataset. The graph shows that the revenue and adjusted revenue of movies has increased over time. The adjusted revenue takes into account the inflation over time and is a better representation of the true value of the revenue generated by the movies. The graph also shows that the revenue of movies has increased at a faster rate than the inflation rate.

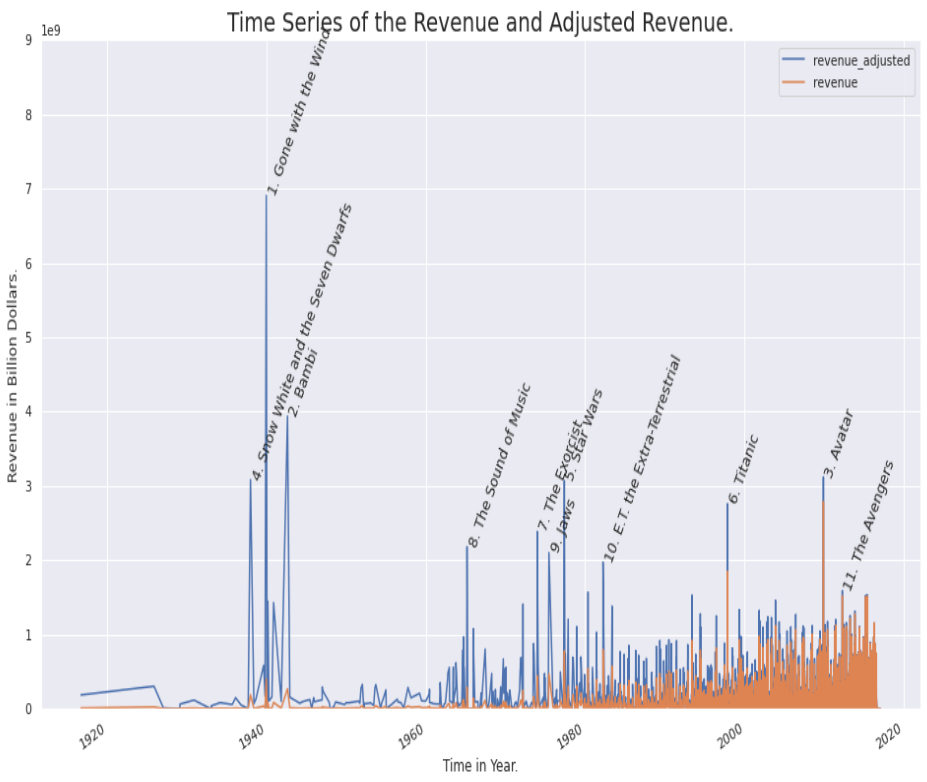


Figure 2.15: Movies Revenue and Adjusted Revenue vs Movie Year

14. **Adjusted - Movie Revenue vs Movie Budget**: A scatter plot of movie revenue vs budget adjusted to inflation. The plot shows a positive correlation between the two variables, with some outliers. The x-axis represents the adjusted movie budget in 100 million dollars, while the y-axis represents the adjusted movie revenue in billion dollars. The plot shows that movies with higher budgets tend to have higher revenues. However, there are some movies that have much higher revenues than expected based on their budgets. These movies are labeled as outliers in the plot. The outliers include “Star Wars: The Force Awakens”, “Jurassic World”, “Furious 7”, “The Avengers”, “Harry Potter and the Deathly Hallows: Part 2”, “Transformers: Dark of the Moon”, “The Dark Knight Rises”, “Pirates of the Caribbean: On Stranger Tides”, “Toy Story 3”, and “Alice in Wonderland”.

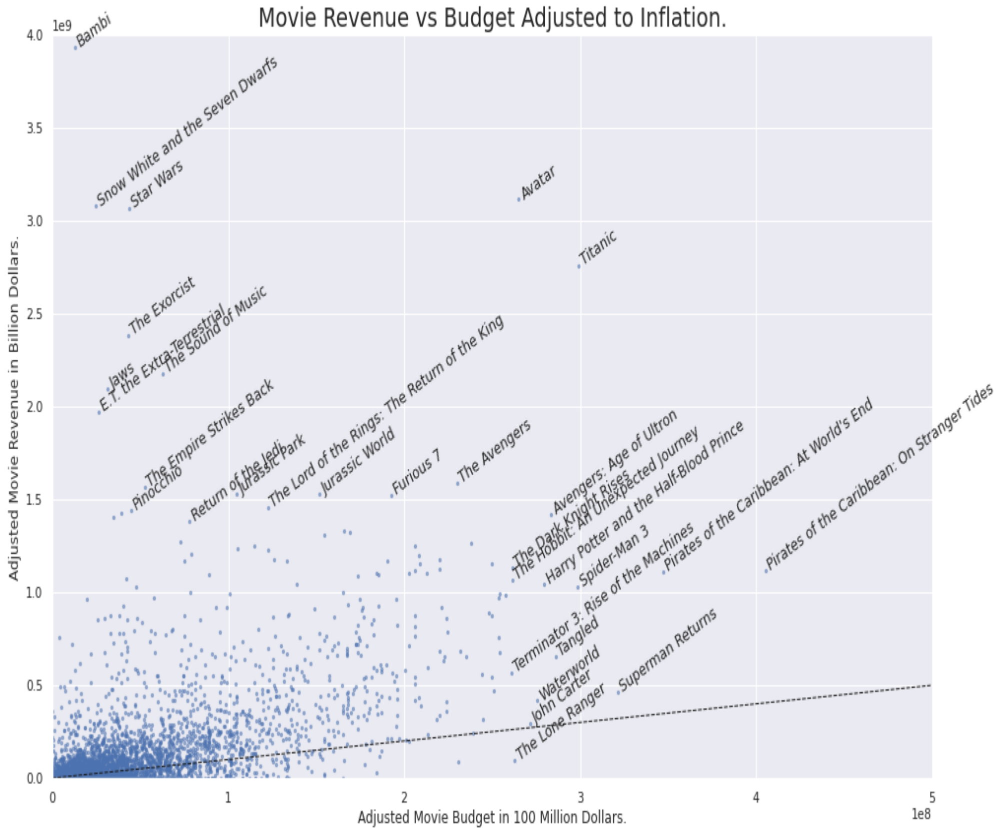


Figure 2.16: Adjusted - Movie Revenue vs Movie Budget

15. **Adjusted Movies Revenue and Release Year with Genres**: The graph shows data from 1990 to 2015. The x-axis represents the release year, while the y-axis represents the movie's revenue in dollars. The bars are color-coded by genre, with each color representing a different genre. The graph shows that the highest revenue appears to be generated by movies in the Action, Adventure, and Animation genres. These genres have consistently generated higher revenue than other genres over the years. The Drama genre also appears to be popular, with high revenue in the 1990s and 2000s. The Comedy genre has been consistently popular over the years, but its revenue has been lower than that of the aforementioned genres. The Science Fiction genre has also been consistently popular, with high revenue in the 1990s and 2010s.

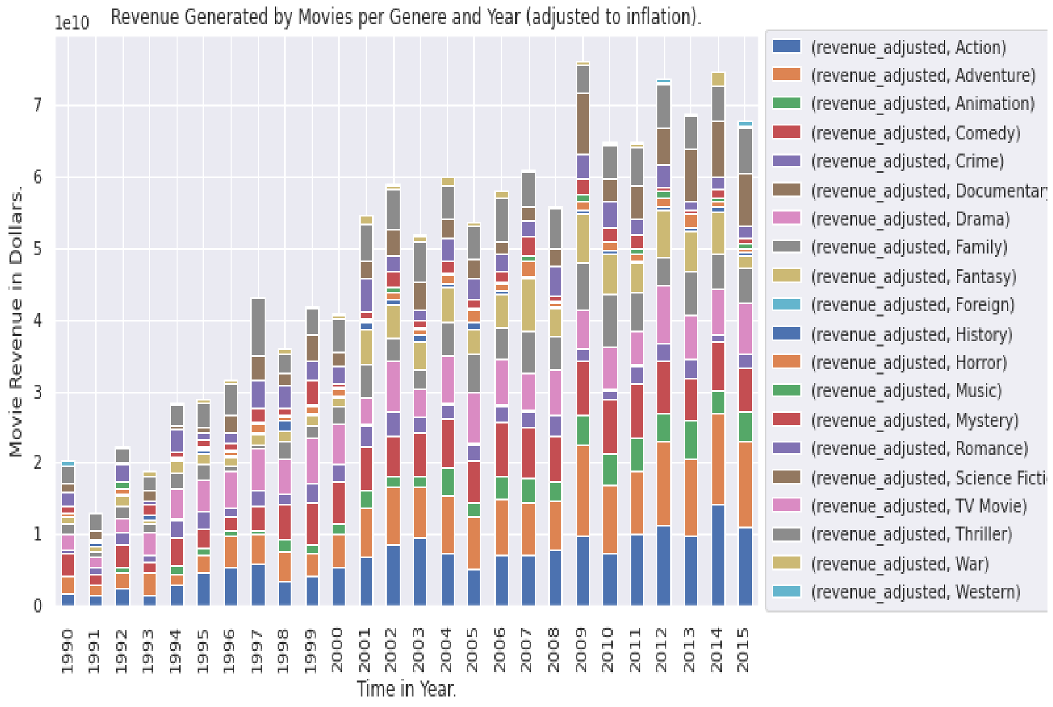


Figure 2.17: Adjusted Movies Revenue and Release Year with Genres

16. **Average Movie Rating vs Movie Revenue**: The x-axis represents the movie revenue, while the y-axis represents the average movie rating. The data points are colored according to the decade in which the movie was released, with blue representing the 1920s and purple representing the 2010s. The plot appears to show that movies with higher ratings tend to have higher revenues. There is a positive correlation between the two variables, with some outliers. The plot also shows that movies released in the 2010s have higher revenues than movies released in previous decades.

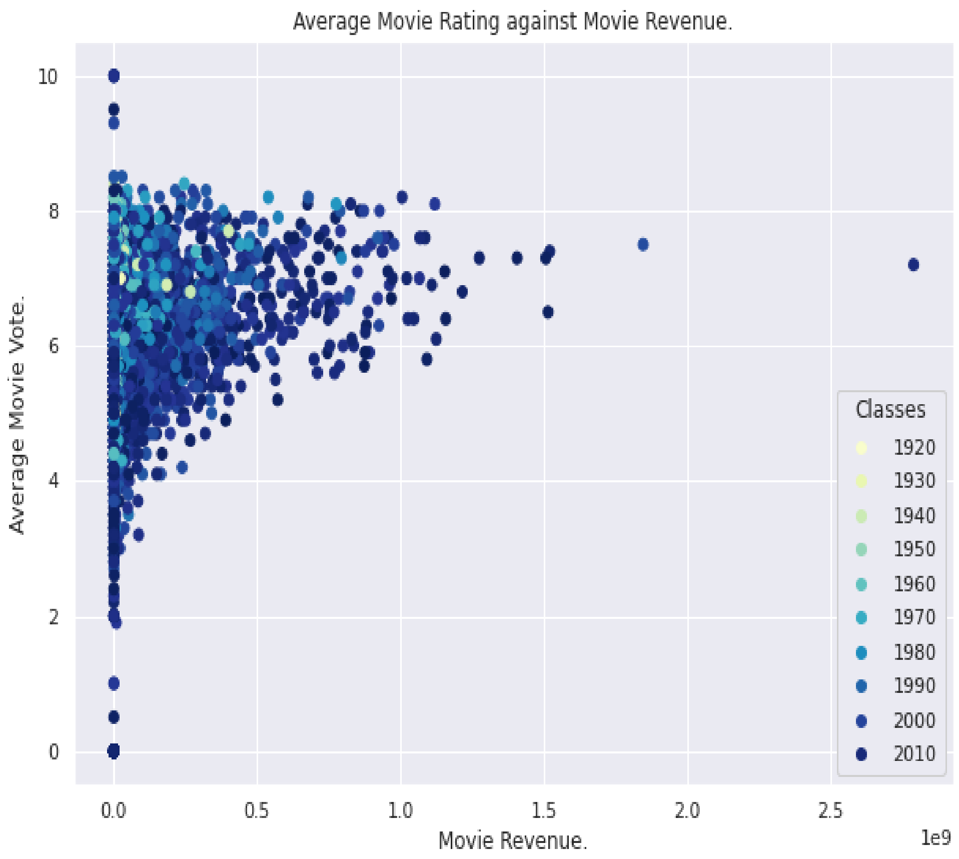


Figure 2.18: Average Movie Rating vs Movie Revenue

17. **Movie Rating**: A bar graph that shows the distribution of the number of movies by the number of votes. The x-axis represents the number of votes, while the y-axis represents the number of movies. The bars decrease in height as the number of votes increases. The highest bar is at around 2000 votes, with around 200 movies. The lowest bar is at around 14000 votes, with around 0 movies.

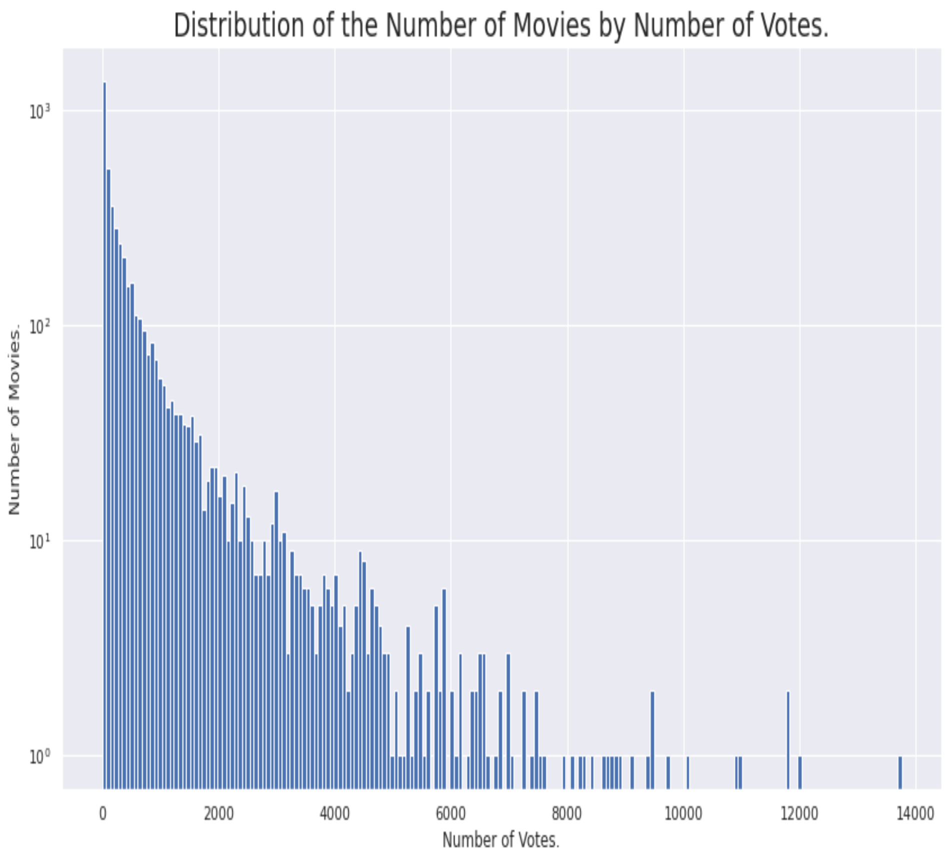


Figure 2.19: Movie Rating

18. **Production Companies vs Number of Movies**: The graph indicates that there are a large number of production companies that have produced a small number of movies, and a smaller number of production companies that have produced a larger number of movies. The bars on the left side of the graph are taller than the bars on the right side of the graph. The highest bar on the graph is on the left side of the graph and it appears to be around 100 production companies that have produced around 10 movies.

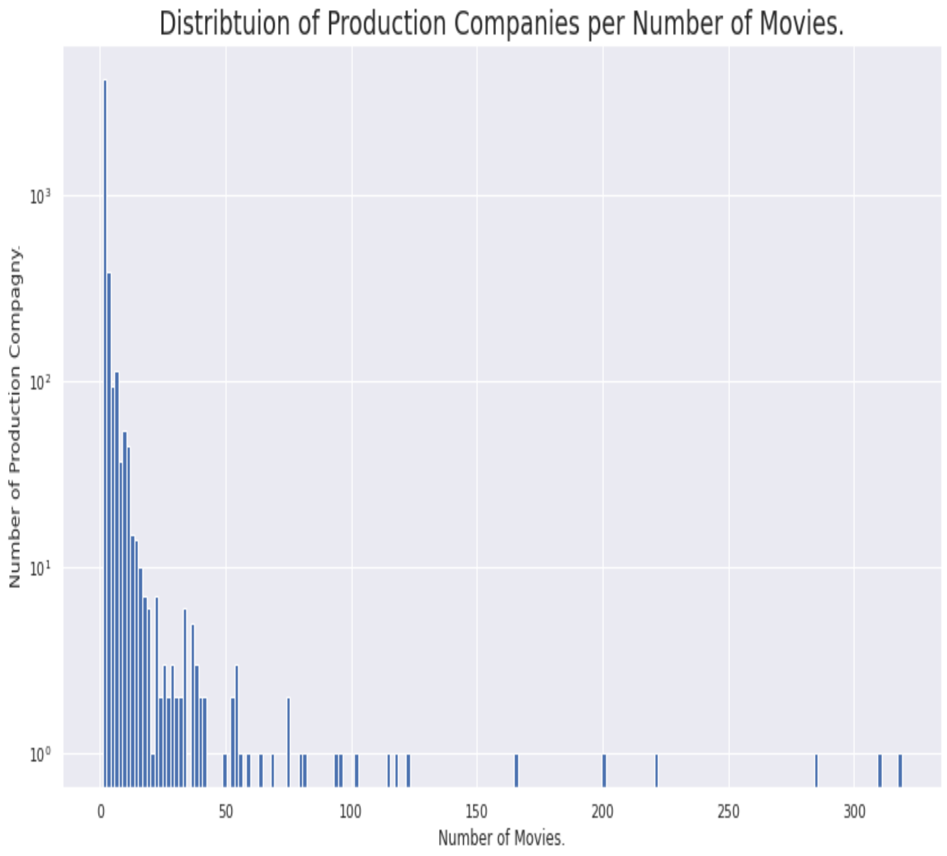


Figure 2.20: Production Companies vs Number of Movies

19. **Top 20 Production Companies**: a bar graph that shows the top 20 movie production companies and the number of movies they have produced according to the TMDB 5000 movie dataset.

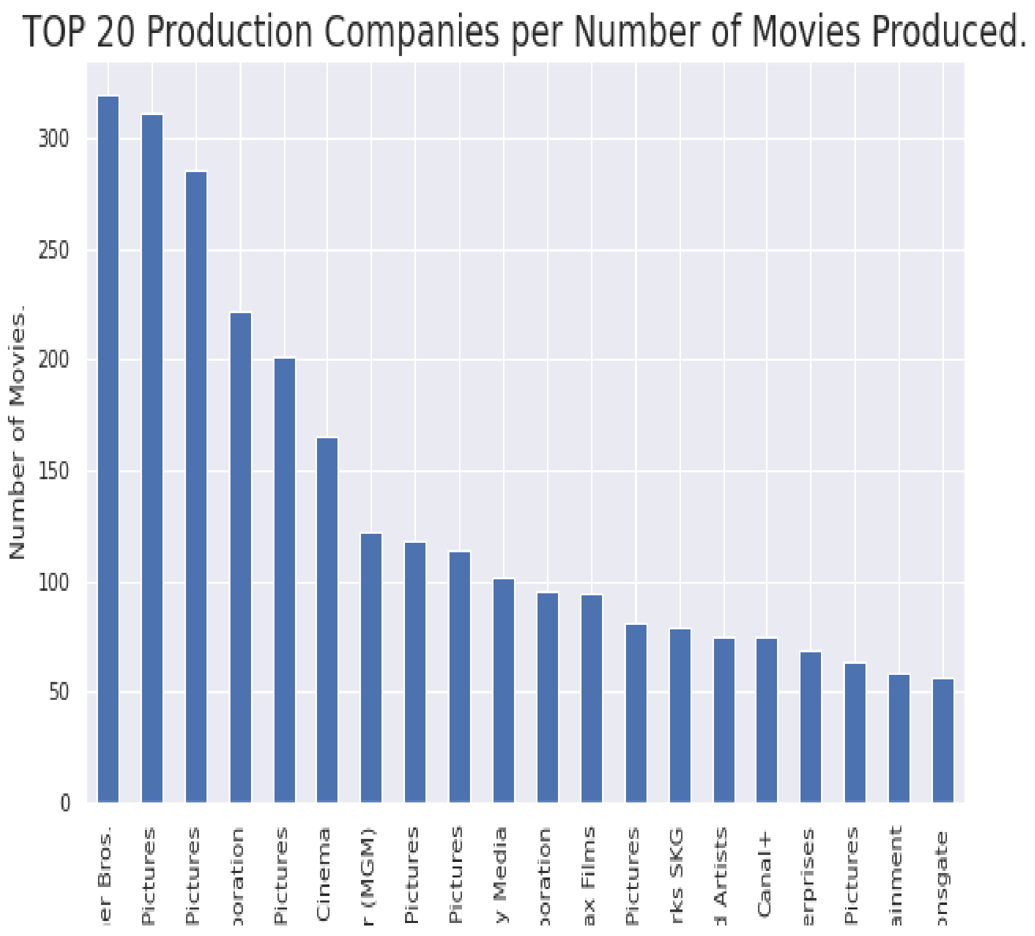


Figure 2.21: Top 20 Production Companies

20. **Cumulative Revenues of Top 20 Production Companies**: a bar graph that represents the top 20 movie production companies with cumulative revenue. The x-axis of the graph represents the production companies, and the y-axis represents the revenue in billions of dollars.

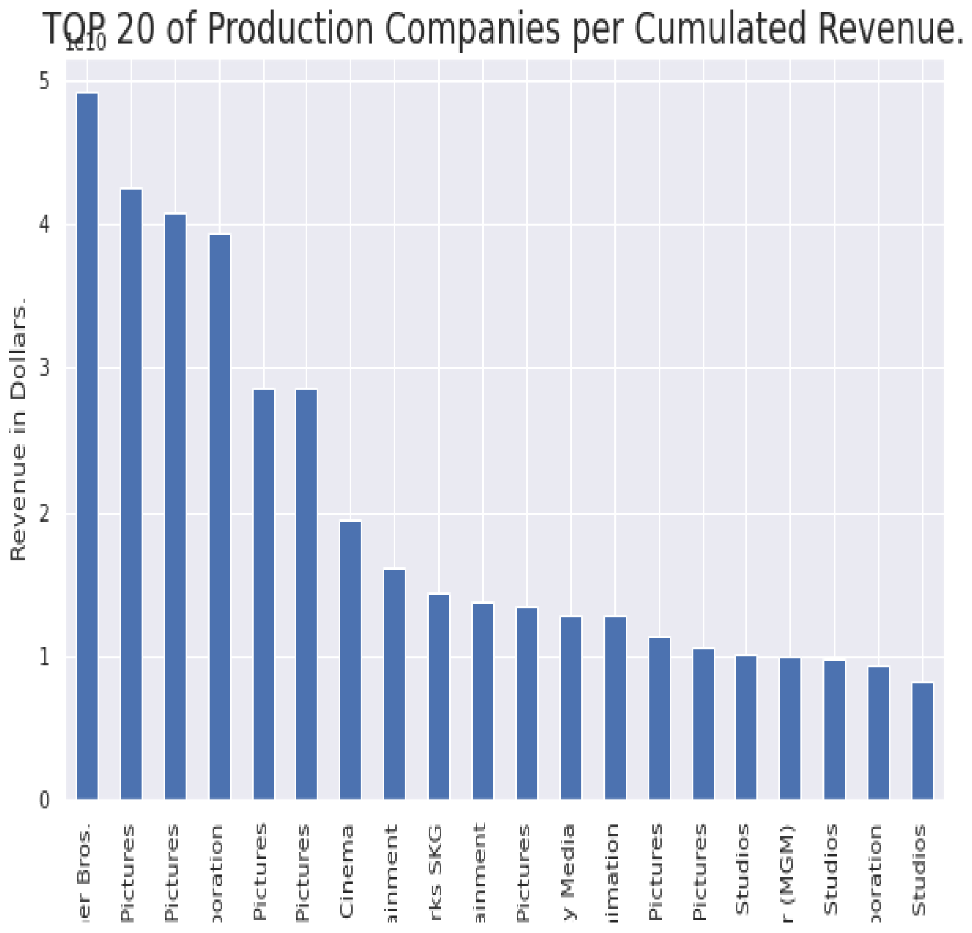


Figure 2.22: Cumulative Revenues of Top 20 Production Companies

21. **Average Revenues of Top 20 Production Companies**: a bar graph that shows the top 20 production companies per average revenue. The x-axis of the graph represents the production companies, and the y-axis represents the average revenue in dollars.

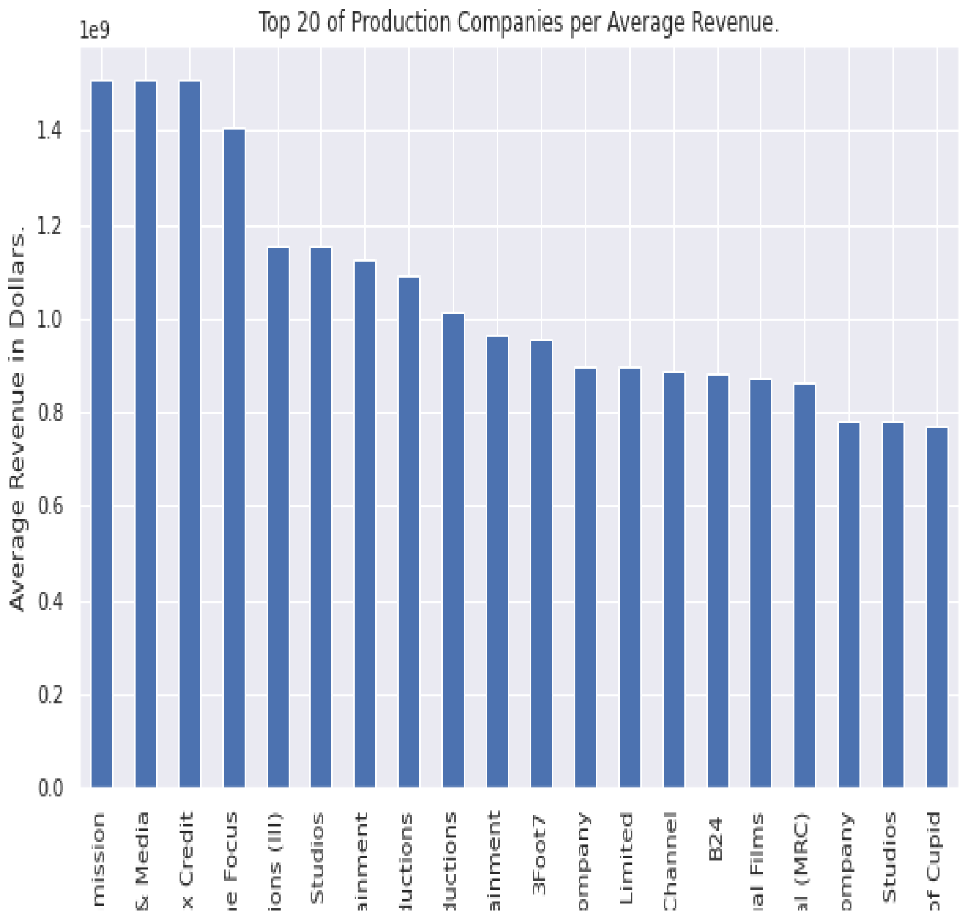


Figure 2.23: Average Revenues of Top 20 Production Companies

22. **Gender vs Mean Revenue**: a bar graph that shows the mean movie revenue earned by each gender of the tmdb 5000 movie dataset. The x-axis is labeled "Gender" and the y-axis is labeled "Mean Revenue in Billion Dollars". The two bars represent the mean revenue earned by females and males respectively. The bar for females is slightly shorter than the bar for males.

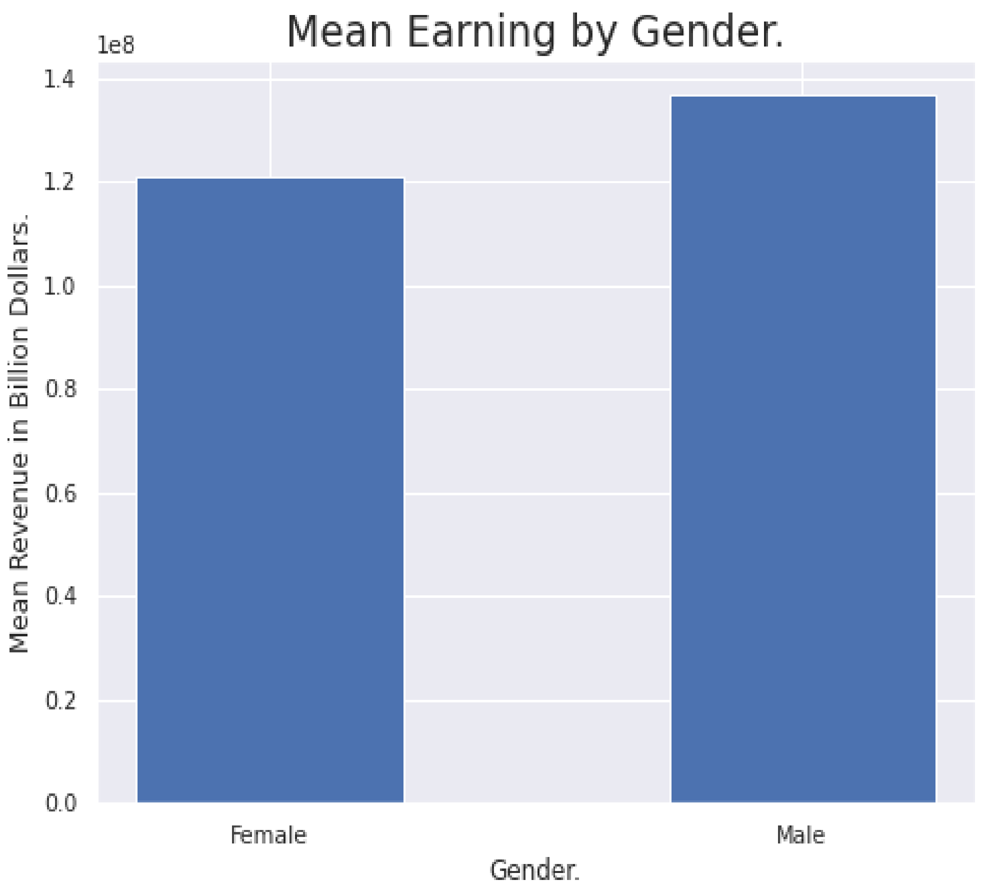


Figure 2.24: Gender vs Mean Revenue

23. **Year vs Mean Revenue w.r.t gender**: a line graph that shows the mean revenue in millions of dollars generated by each gender with time of the TMDB 5000 movie dataset. The graph appears to show that the revenue generated by each gender has increased over time. The mean revenue generated by male actors has been consistently higher than that generated by female actors throughout the years. However, the difference between the two has decreased over time.

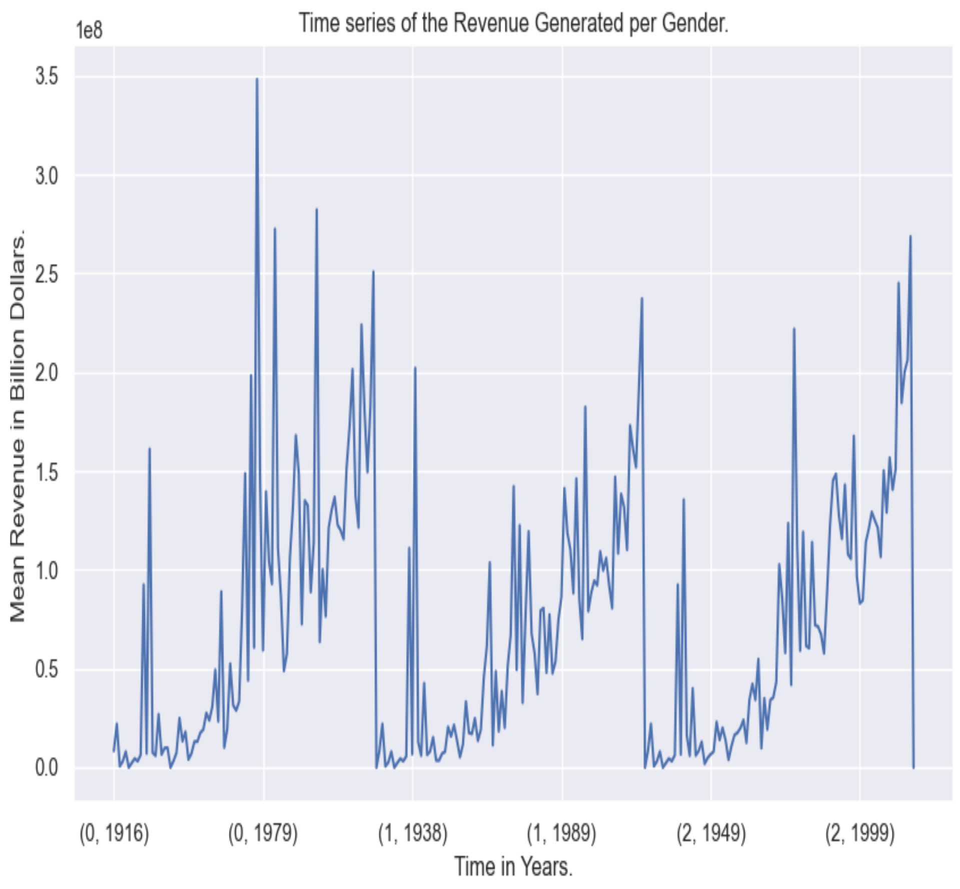
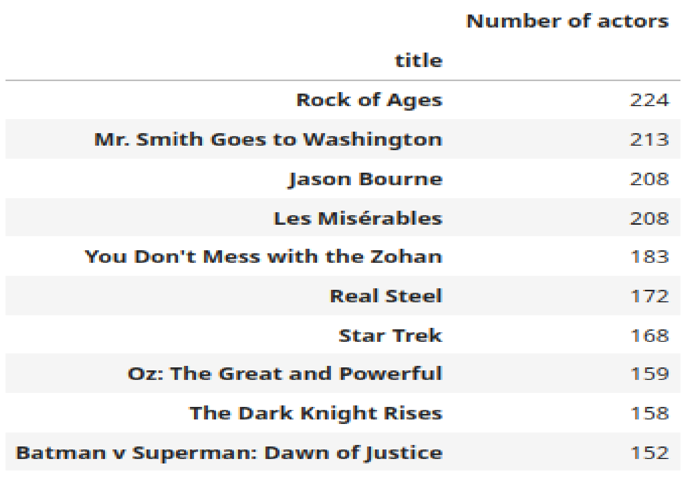


Figure 2.25: Year vs Mean Revenue w.r.t gender

24. A table of the top movies with the largest star cast of TMDB 5000 movies dataset. The table has two columns: title and number of actors. The table is sorted by the number of actors in descending order.



	Number of actors
title	
Rock of Ages	224
Mr. Smith Goes to Washington	213
Jason Bourne	208
Les Misérables	208
You Don't Mess with the Zohan	183
Real Steel	172
Star Trek	168
Oz: The Great and Powerful	159
The Dark Knight Rises	158
Batman v Superman: Dawn of Justice	152

Figure 2.26: Top 10 Movies with more No. of Actors

25. **Number of Movies vs Number of Actors**: The x-axis represents the number of actors and the y-axis represents the number of movies. The histogram shows that most movies have between 0 and 50 actors, with a few movies having more than 150 actors. The highest number of movies have between 0 and 10 actors. The histogram has a long tail, with a few movies having a large number of actors.

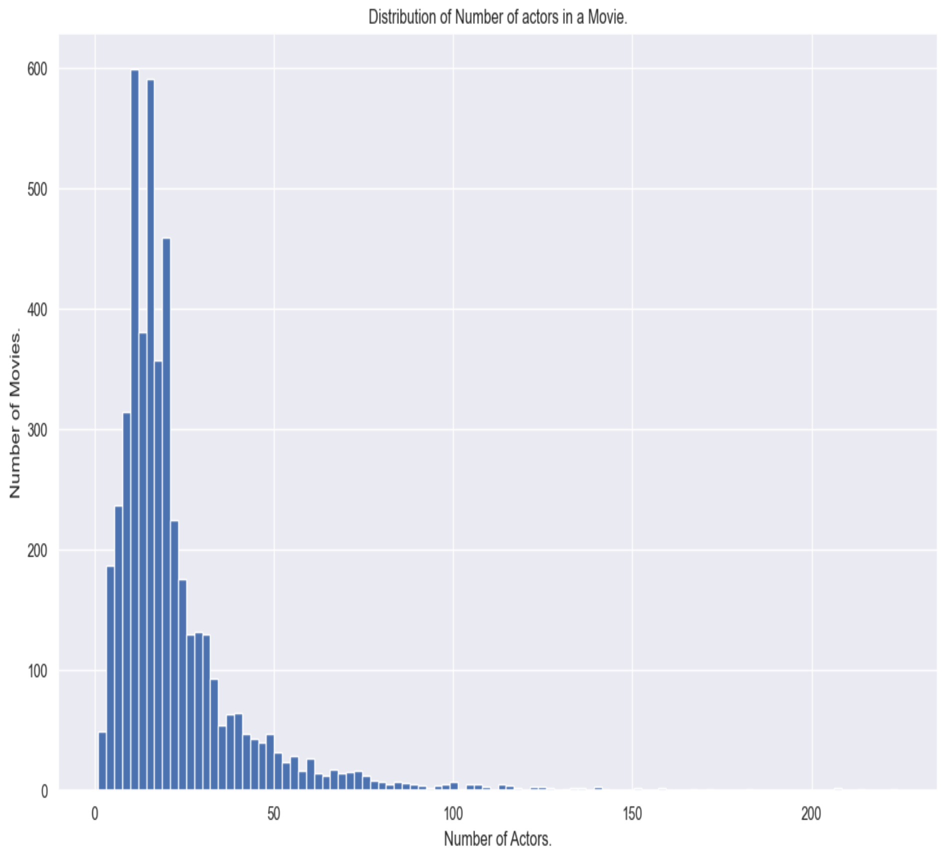


Figure 2.27: Number of Movies vs Number of Actors

26. **Number of Movies vs Actors**: This barplot shows the top 20 actors with the highest number of movies delivered. The x-axis shows the actors' names and the y-axis shows the number of movies delivered. The actor with the highest number of movies delivered is "D. Willis" with around 68 movies, followed by "A. Pacino" with around 64 movies.

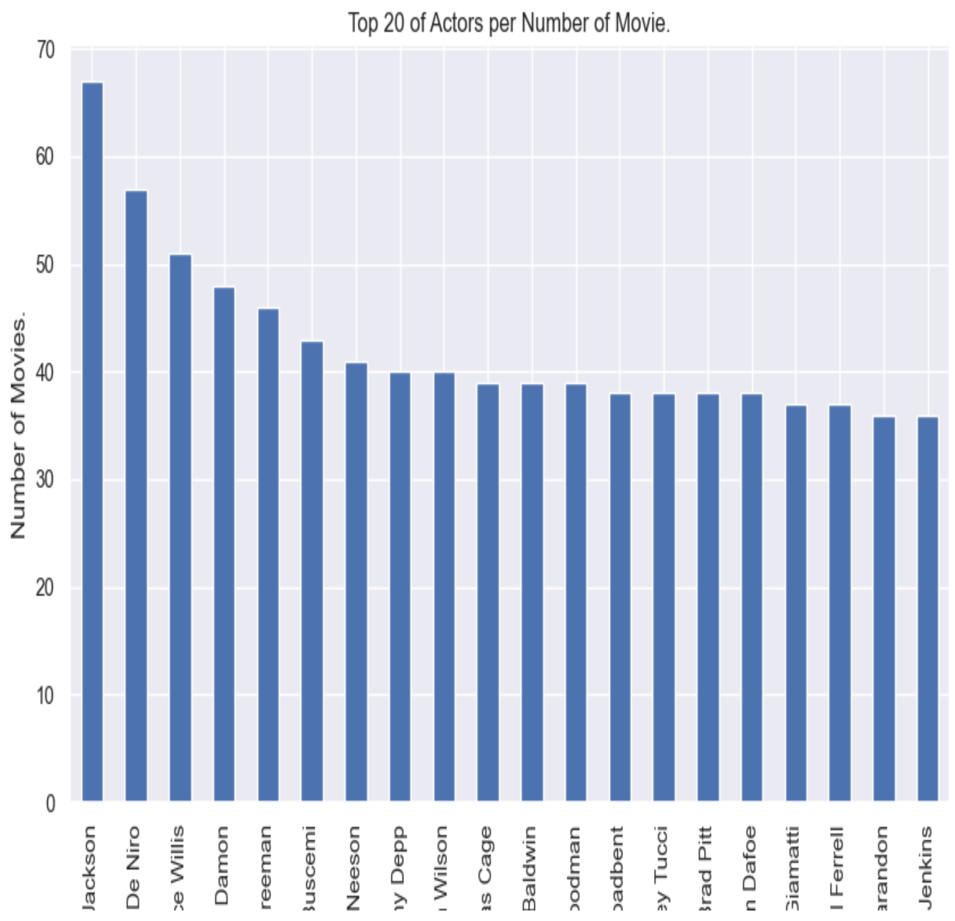


Figure 2.28: Number of Movies vs Actors

27. **Revenue Generated vs Actors**: This is a bar plot of the top 20 actors with the highest number of revenue generated. The x-axis shows the names of the actors, and the y-axis shows the amount of revenue generated in millions of dollars. The bars are arranged in descending order of revenue generated.

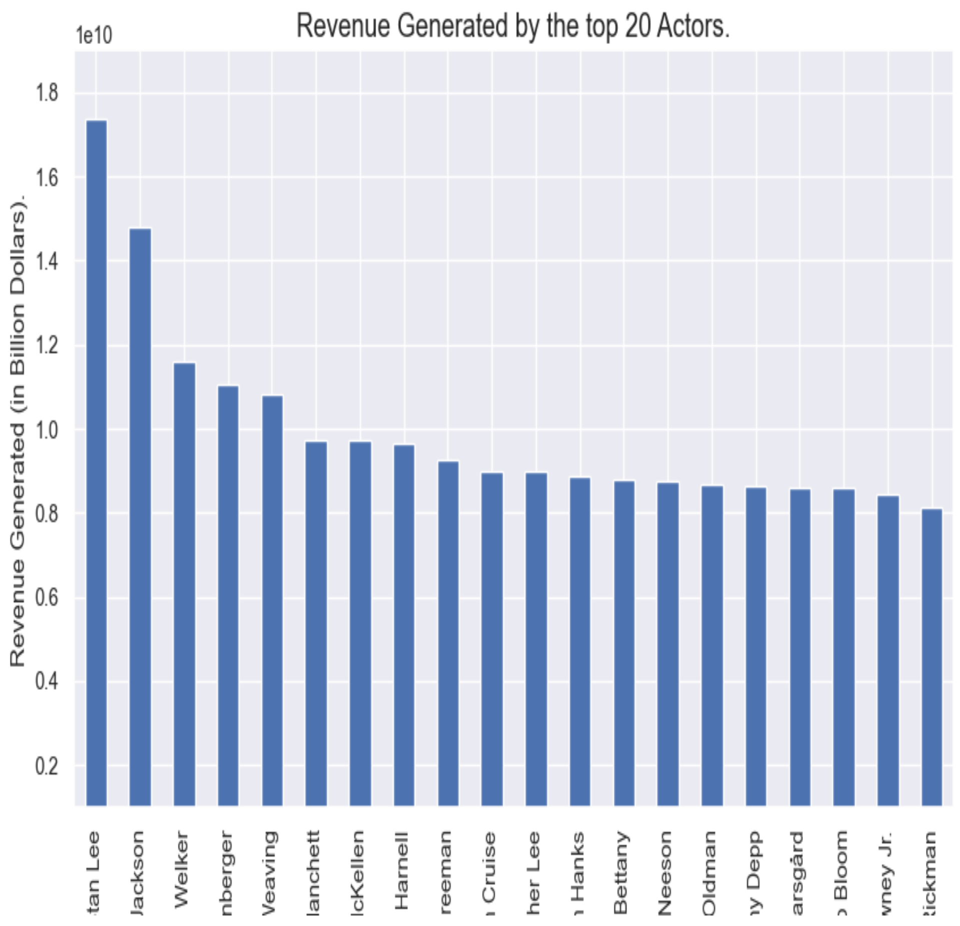
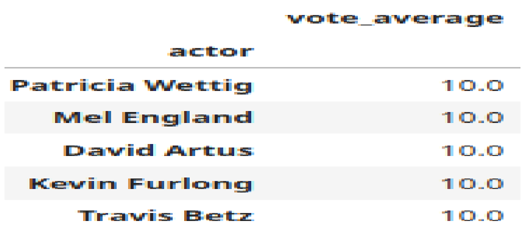


Figure 2.29: Revenue Generated vs Actors

28. Grouping of dataset w.r.t each actor on the basis of average voting and finding the top 5 actors with highest votes acquired.



	vote_average
actor	
Patricia Wettig	10.0
Mel England	10.0
David Artus	10.0
Kevin Furlong	10.0
Travis Betz	10.0

Figure 2.30: Top 5 Actors with more Vote Count

29. **Average Score vs Actors**: A histogram that shows the distribution of average ratings for actors. The x-axis represents the average score, ranging from 0 to 10. The y-axis represents the number of actors, ranging from 0 to 8000. The highest bar is around the 6-7 range, indicating that the majority of actors have an average rating between 6 and 7. The bars decrease in height as the average rating increases or decreases from the 6-7 range.

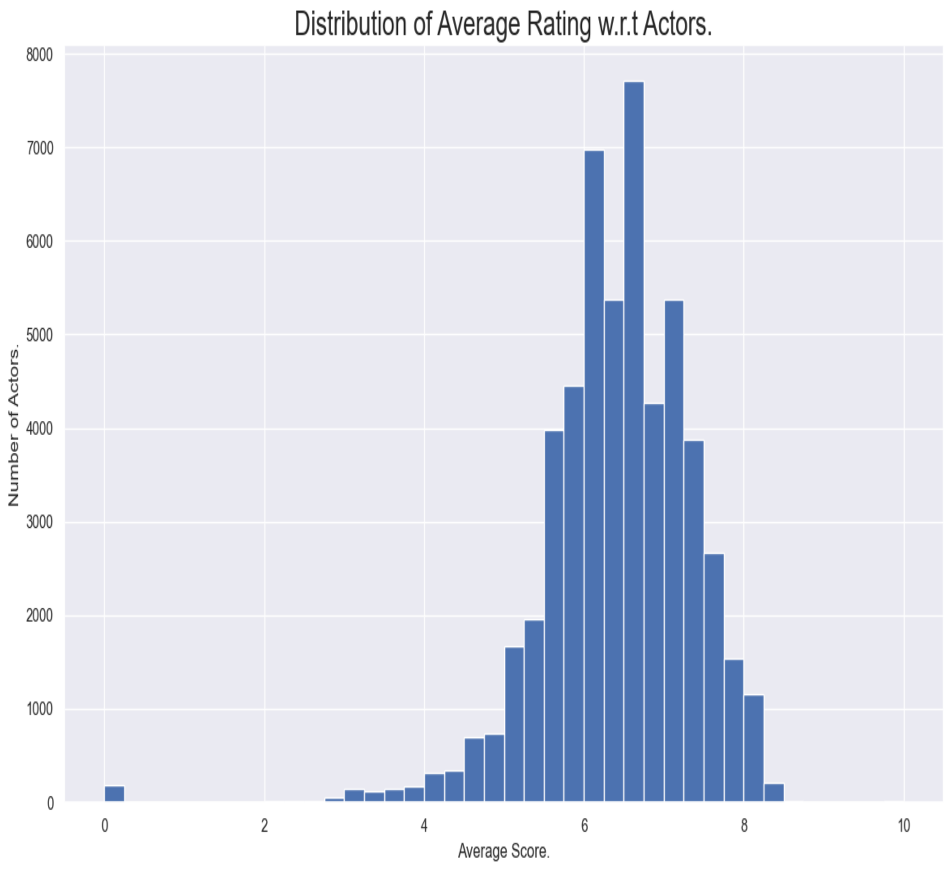
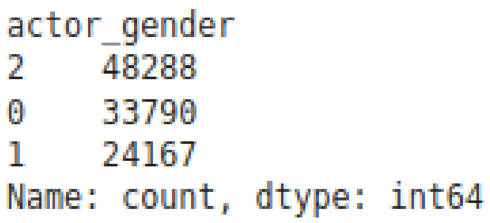


Figure 2.31: Average Score vs Actors

30. Calculating the total number of people of specific gender working in this industry. So 0 is non-specified, 1 is Female, and 2 is Male.



```
actor_gender
2    48288
0    33790
1    24167
Name: count, dtype: int64
```

Figure 2.32: Total actor of each Gender

31. **Movie Revenue vs Number of Actors**: This is a scatterplot of the relationship between the number of actors in a movie and the movie's revenue. The plot shows that there is a positive correlation between the number of actors and the movie's revenue. However, there are a few outliers that do not follow this trend. For example, "Jurassic World" and "Furious 7" have a high number of actors but a lower revenue than expected. "Star Wars:

The Force Awakens" and "Jurassic Park" have a lower number of actors but a higher revenue than expected.

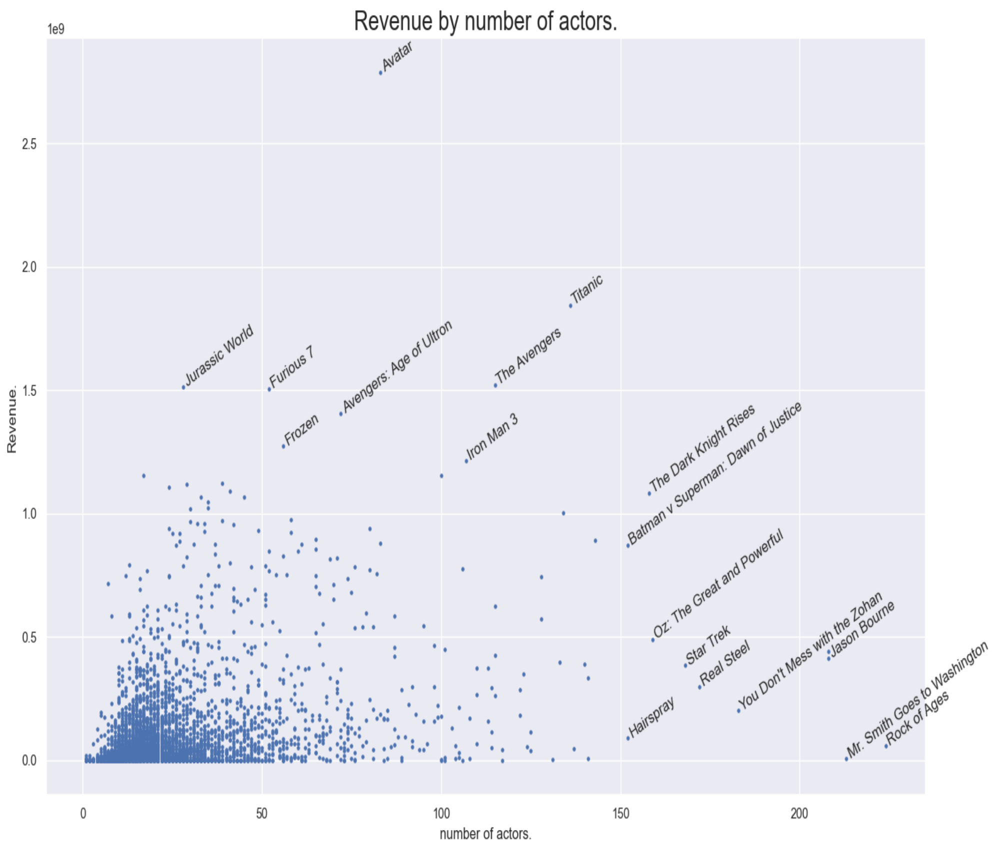
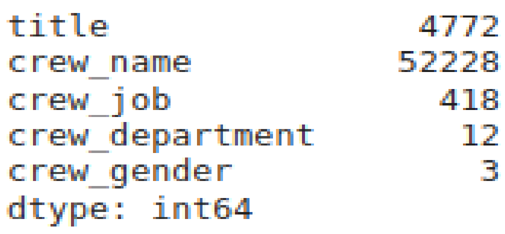


Figure 2.33: Movie Revenue vs Number of Actors

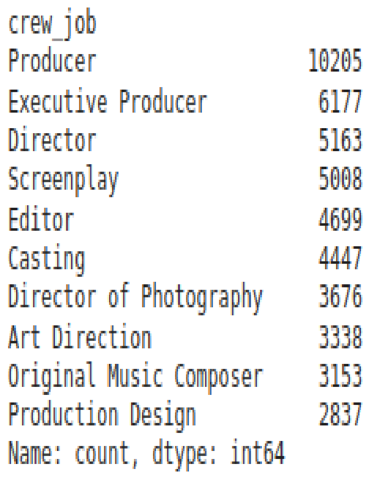
32. Summing the total unique components of details given in crew data.



```
title                4772
crew_name           52228
crew_job              418
crew_department        12
crew_gender             3
dtype: int64
```

Figure 2.34: Crew Details

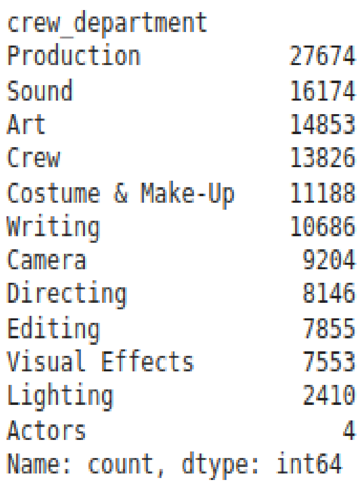
33. Calculating the total number of each crew job. The table has 9 rows and 2 columns. The first column is the crew job title and the second column is the total number of each job.



```
crew_job
Producer                   10205
Executive Producer          6177
Director                    5163
Screenplay                  5008
Editor                      4699
Casting                     4447
Director of Photography     3676
Art Direction               3338
Original Music Composer     3153
Production Design           2837
Name: count, dtype: int64
```

Figure 2.35: Total No. of people with respective Crew Job

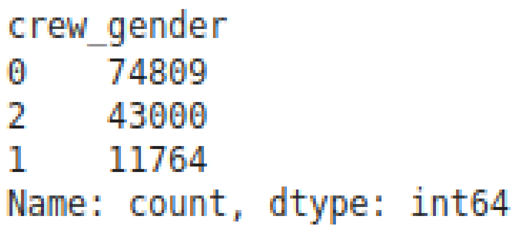
34. Calculating the total number of each crew department.



```
crew_department
Production          27674
Sound               16174
Art                 14853
Crew                13826
Costume & Make-Up   11188
Writing             10686
Camera               9204
Directing            8146
Editing              7855
Visual Effects       7553
Lighting             2410
Actors                  4
Name: count, dtype: int64
```

Figure 2.36: Crew Departmets with No. of People working in it

35. Calculating the total number of each crew gender.

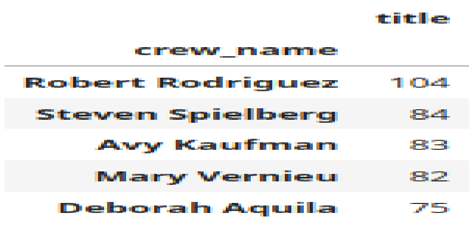


```
crew_gender
0    74809
2    43000
1    11764
Name: count, dtype: int64
```

Figure 2.37: Total crew of each Gender

36. Calculating the number of movies done by each crew member and mentioning the the top

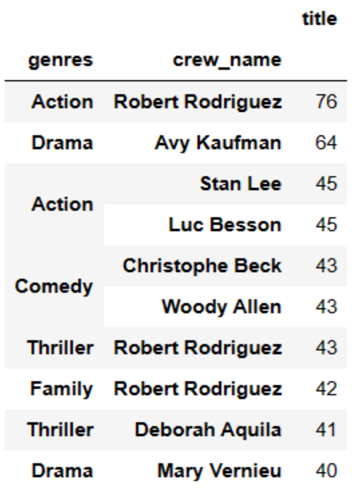
5 crew members with more number of movies delivered.



	title
crew_name	
Robert Rodriguez	104
Steven Spielberg	84
Avy Kaufman	83
Mary Vernieu	82
Deborah Aquila	75

Figure 2.38: Top 5 Crew Members with more Movies

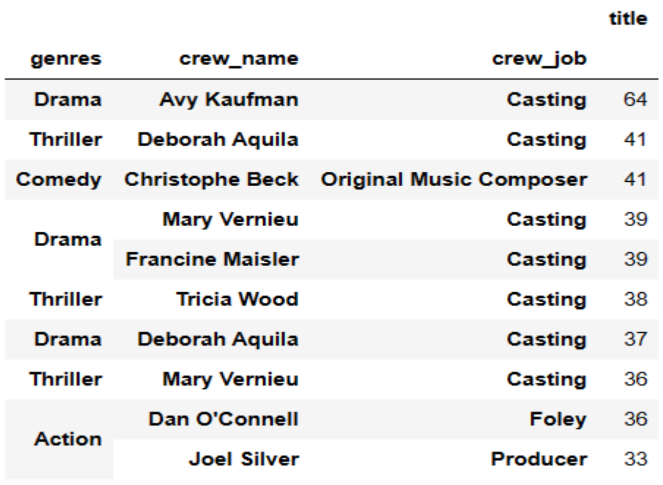
37. Grouping of each crew member with title into their respective genres to find out the top 10 crew members with more number of movies done and in which genres.



		title
genres	crew_name	
Action	Robert Rodriguez	76
Drama	Avy Kaufman	64
Action	Stan Lee	45
	Luc Besson	45
Comedy	Christophe Beck	43
	Woody Allen	43
Thriller	Robert Rodriguez	43
Family	Robert Rodriguez	42
Thriller	Deborah Aquila	41
Drama	Mary Vernieu	40

Figure 2.39: Top 10 Crew with respective genre

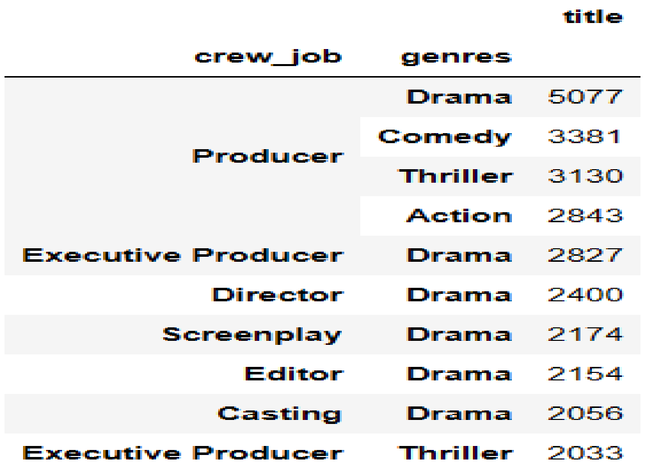
38. Exploding the above dataset into 'genres', 'crew-name', 'crew-job', 'title' to find out in which department the top 10 crew members have done their job.



			title
genres	crew_name	crew_job	
Drama	Avy Kaufman	Casting	64
Thriller	Deborah Aquila	Casting	41
Comedy	Christophe Beck	Original Music Composer	41
Drama	Mary Vernieu	Casting	39
	Francine Maisler	Casting	39
Thriller	Tricia Wood	Casting	38
Drama	Deborah Aquila	Casting	37
Thriller	Mary Vernieu	Casting	36
Action	Dan O'Connell	Foley	36
	Joel Silver	Producer	33

Figure 2.40: Top 10 crew members with most crew job and title

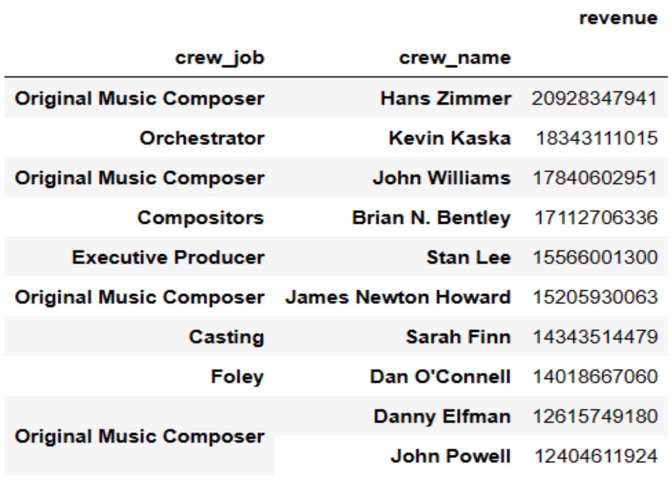
39. Segragating the dataset 'genres', 'title' into grouping of 'crew-job', carving out top 10 of them. This will give us an idea about which crew job has importance in a specific genre.



		title
crew_job	genres	
Producer	Drama	5077
	Comedy	3381
	Thriller	3130
	Action	2843
Executive Producer	Drama	2827
Director	Drama	2400
Screenplay	Drama	2174
Editor	Drama	2154
Casting	Drama	2056
Executive Producer	Thriller	2033

Figure 2.41: Top 10 genres with most particular crew jobs

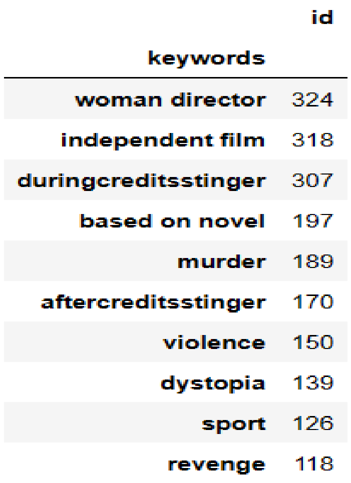
40. Grouping of crew member w.r.t their job and total revenue earned by them respectively. Here is the top 10 results of it.



		revenue
crew_job	crew_name	
Original Music Composer	Hans Zimmer	20928347941
Orchestrator	Kevin Kaska	18343111015
Original Music Composer	John Williams	17840602951
Compositors	Brian N. Bentley	17112706336
Executive Producer	Stan Lee	15566001300
Original Music Composer	James Newton Howard	15205930063
Casting	Sarah Finn	14343514479
Foley	Dan O'Connell	14018667060
Original Music Composer	Danny Elfman	12615749180
	John Powell	12404611924

Figure 2.42: Top 10 crew w.r.t Revenue

41. Calculating the top 10 Number of times a particular keyword is used in movies.



	id
keywords	
woman director	324
independent film	318
duringcreditsstinger	307
based on novel	197
murder	189
aftercreditsstinger	170
violence	150
dystopia	139
sport	126
revenge	118

Figure 2.43: Top 10 Keywords

42. **Production Countries**: Barplot of top 20 production countries. Since the complete dataset belongs to Hollywood movies only. So it is obvious that very large chunk of movies will be produced in US or UK. The other may be of hollywood but produced in different countries.

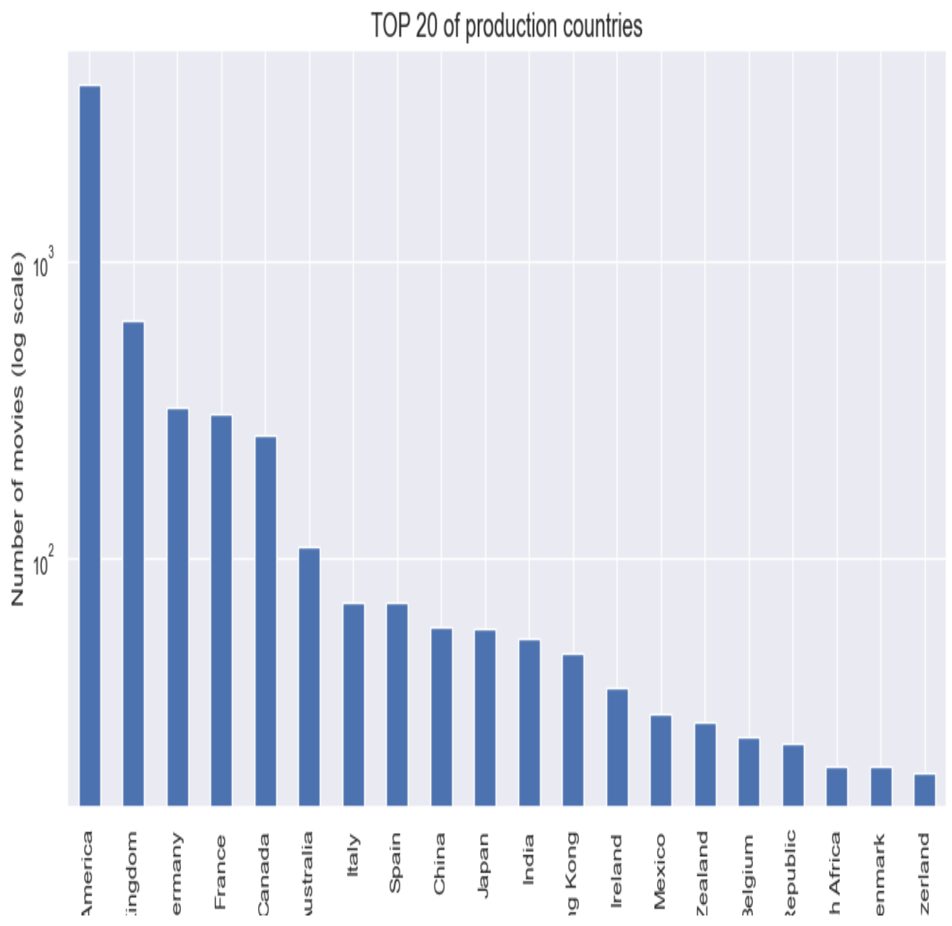
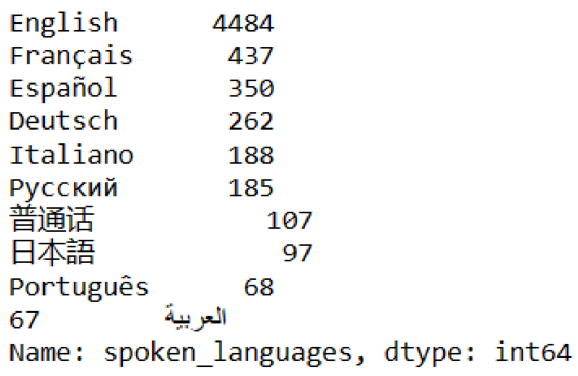


Figure 2.44: Production Countries

43. Calculating the total Number of movies made or dubbed in each language.



```
English       4484
Français       437
Español        350
Deutsch        262
Italiano       188
Русский        185
普通话             107
日本語              97
Português       68
67         العربية
Name: spoken_languages, dtype: int64
```

Figure 2.45: Total Movies in each language

44. **Spoken Languages**: Barplot of top 20 spoken languages of the movies with x-axis as language and y-axis as the number of movies.

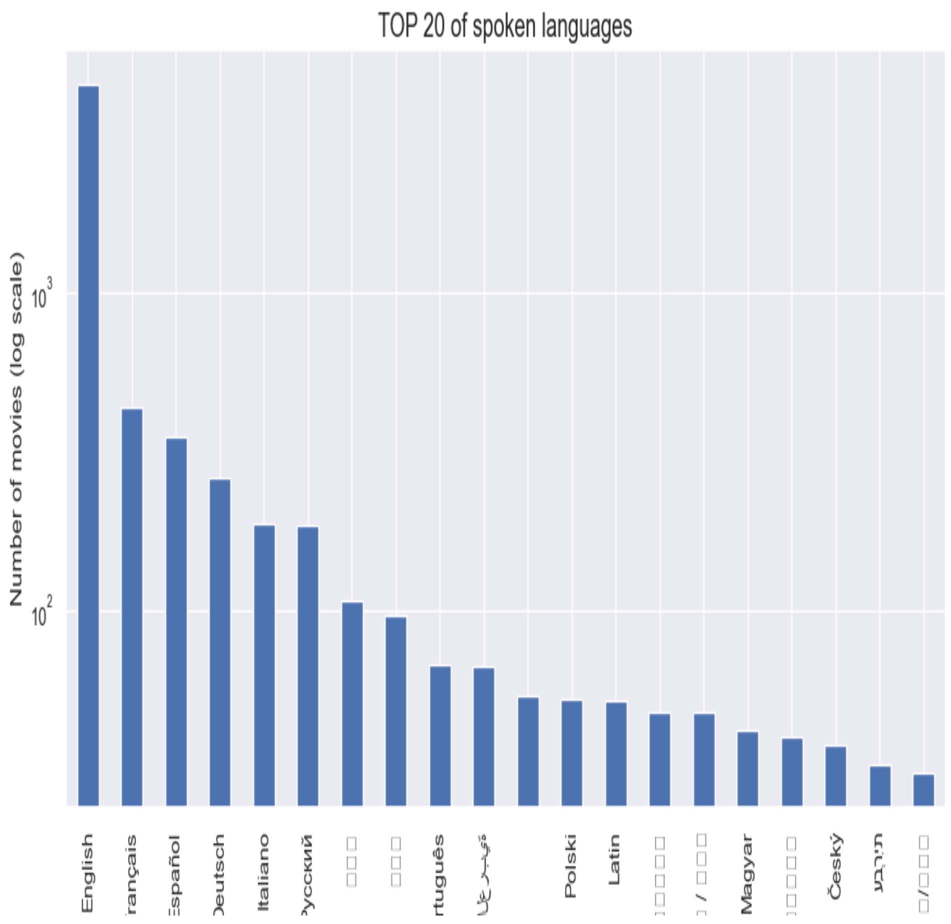


Figure 2.46: Spoken Languages

45. **Original Languages**: Barplot of top 20 original languages of the movies with x-axis as original languages and y-axis as number of movies.

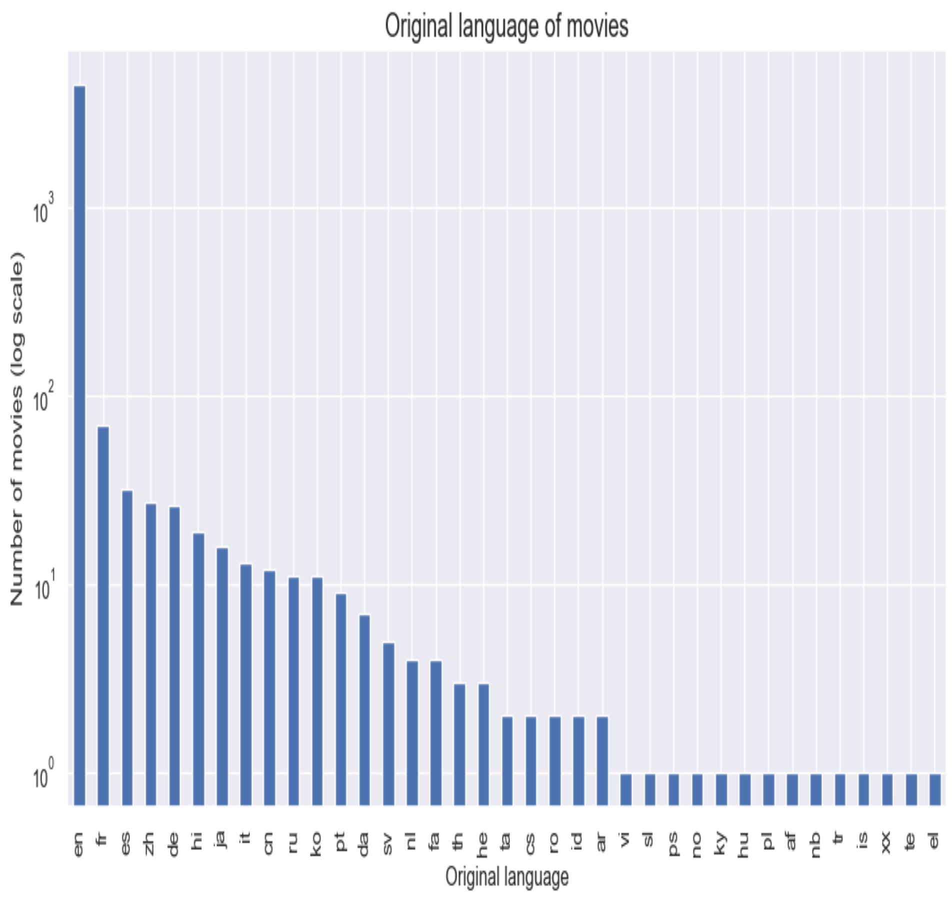
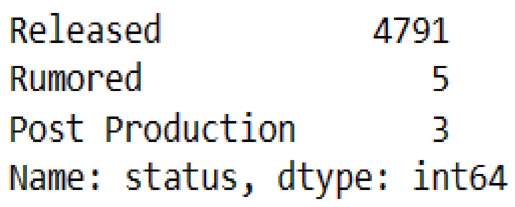


Figure 2.47: Original Languages

46. Calculating the total Number of movies w.r.t their status. Since there are very few number of movies which did not released or never made.



```
Released            4791
Rumored                5
Post Production        3
Name: status, dtype: int64
```

Figure 2.48: Movies Status

Chapter 3

Hybrid Movie Recommender Model

3.1 Introduction

In this project, I am introducing a hybrid movie recommender system that combines two powerful recommendation techniques: collaborative-based filtering and content-based filtering. The primary objective of this system is to offer personalized movie recommendations to users based on the movies they have recently watched.

3.1.1 Collaborative-Based Filtering

This method identifies users with similar movie preferences by analyzing their interactions with movies, such as ratings and reviews. By connecting users with like-minded peers, this approach recommends movies based on the preferences of similar users, introducing serendipity into the recommendations.

3.1.2 Content-Based Filtering

Content-based filtering evaluates the inherent characteristics of movies, i.e keywords. It uses these attributes to suggest movies that align thematically or stylistically with the user's recent viewing choices.

The implementation includes data collection and preprocessing, collaborative filtering, content-based filtering, and a hybridization step to provide well-rounded and diverse movie suggestions. This approach enhances personalization, introduces serendipitous movie discoveries, and ensures thematic consistency in recommendations.

3.1.3 Methodology for Movie Selection and Filtering

The process of selecting and filtering movies is a critical step in the development of our hybrid movie recommender system. To provide users with highly relevant and engaging movie recommendations, we employ a multi-stage filtering approach. This methodology ensures that the movies recommended to users are not only similar to their recently watched movies but also of high quality and aligned with their preferences.

Keyword, Tagline, and Genres Filtering

The first stage of our methodology involves filtering the top 100 similar movies based on key attributes such as "keywords," "tagline," and "genres." These attributes provide valuable insights into the content, theme, and genre of the movies.

Vote Count Selection

After obtaining the top 100 movies from the initial filter, we apply a secondary filter based on their "vote-count." This filter allows us to identify movies that have garnered substantial attention and engagement from the audience. We select the top 50 movies based on their vote counts for further analysis.

Overview-Based Filtering

In the final filtering process, we take the top 50 movies from the previous step and perform a detailed analysis. Specifically, we focus on the "overview" of these movies. By considering the content and narrative descriptions in the overviews, we identify the top 5 similar movies. These top 5 movies are the final recommendations that will be presented to the user.

Rationale

1. The initial keyword, tagline, and genres filter ensures that we capture movies with thematic and genre similarities to the user's preferences.
2. The vote count filter takes into account the popularity and general audience engagement with the movies. It helps us prioritize movies that have received significant attention and positive feedback.
3. The overview-based filtering allows us to fine-tune the recommendations by considering the narrative content of the movies. This step enhances the quality and relevance of the final recommendations.

Our multi-stage movie selection and filtering methodology is designed to curate a list of top 5 similar movies for users, ensuring that the recommendations are not only tailored to their preferences but also represent high-quality and engaging cinematic choices. This approach aligns with our goal of delivering a personalized and enriching movie-watching experience to our users.

3.2 Data Preprocessing for Model

3.2.1 Libraries/Functions Required

1. pandas and numpy are libraries for data manipulation and numerical computations in Python.
2. CountVectorizer and TfidfVectorizer are tools for text analysis and feature extraction.
3. cosine-similarity is a function to compute cosine similarity between data points, commonly used in recommendation systems.
4. json is a library for reading and writing JSON data.
5. re is the regular expressions library for pattern matching and text processing.

3.2.2 Data Preprocessing

- The initial step in our analysis involves the import and loading of datasets into memory.
- As a crucial data preprocessing step, we have identified and removed all missing values from the dataset.
- We have tailored our dataset to include specific columns that are instrumental in our movie recommender system. This subset, referred to as "datasets," encompasses essential attributes such as movie title, description, genres, keywords, tagline, user ratings, and vote counts. These attributes serve as the foundation for our recommendation algorithm, allowing us to provide personalized and engaging movie suggestions to our users.
- I'll extract the genres and keywords information from the dataset, which are stored in json format. I'll create a new dataframe for this.
- Next, we will merge them to prepare for the encoding process.

3.3 Hybrid Movie Recommender System

To construct the movie recommender system:

3.3.1 Text Vectorization and Similarity Findings

1. Firstly we need to enable computation, we need to encode the words from merged column as we cannot process data in string format. However, simply encoding each word into numbers may not provide enough information. In this case, we can use the CountVectorizer to encode the data, which will not only enable computation but also provide additional insights into the data. For this purpose, I will use the stop words provided by scikit-learn as the default, which excludes words like 'the', 'and', 'a', 'an', 'in', 'of', 'to', etc.
2. I'll now use consine similarity to find the similarity. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It is often used in natural language processing and information retrieval to measure how similar two documents are based on their content. The value of cosine similarity ranges from -1 to 1, with 1 indicating that the two vectors are identical and 0 indicating that they are completely dissimilar. Higher values of cosine similarity indicate greater similarity between the vectors.
    - The formula for cosine similarity between two vectors A and B can be expressed as:
    - cosine-similarity = (A . B) / (——A—— ——B——)
    - where A . B represents the dot product of A and B.
    - ——A—— and ——B—— represent the magnitudes of A and B, respectively.
3. Now that we have the cosine similarity table for each movie, we can retrieve the movie ID that the user just watched and find its similarity with other movies.

3.3.2 Function for Hybrid Recommender System

1. Now i'll create a function for my hybrid recommender system. Just by using the title of the movie, the recommender system will suggests top 5 movies that user has just watched.

2. Once we obtain the movie index, we can utilize the cosine-similarity matrix to determine the 100 most similar movies to the movie ID that the user watched.
3. Next, we will select the top 50 movies from the 100 movies based on their vote counts. To achieve this, we will use the vote average approach, where the Weighted Rating (WR) is calculated using the formula:
   - Weighted Rating (WR) = (v / (v + m)) * R + (m / (v + m)) * C
   - We will now create a new dataframe for this purpose.
   - C = Mean vote (or rating) across all movies.
   - m = Minimum votes (or ratings) required for the movie to be considered.
   - Function WR = Calculate the weighted average ratings using the provided formula.
4. Sort the values and pick the highest scores to select the top 50 movies for recommendation.
5. Filter the movies based on their content similarity, and for this, we'll be using the movie overview.
6. To ensure that the user's watched movie is among the top 50, we will manually add it if it is not already there, as we need its overview for comparison. It appears that the movie is already included, as indicated by the "true" output.
7. To start with, in this scenario, I need to eliminate any numbers or symbols from the overview sentences and convert them all to lowercase.
8. Let's now proceed with extracting their features. The main difference between CountVectorizer and TfidfVectorizer is that CountVectorizer only counts the occurrences of each word in the document, while TfidfVectorizer considers the frequency of a word across all documents. TfidfVectorizer is better at identifying the most important words in a document or a corpus.
9. The cosine similarity can now be computed between them.
10. Let's now retrieve the movie index that we want to examine from the similarity table. It's worth noting that each row in the table contains the similarities of that particular row to all the available movies in the table

We have obtained the final output of our basic movie recommendation system!. There are various ways to tackle this problem. In this approach, I am finding the movies sequentially based on the previous results.

3.3.3 Hybrid Movie Recommender System Implementation

In this section of our project, we have implemented a hybrid movie recommender system using the Streamlit framework. This system combines collaborative and content-based filtering approaches to suggest the top 5 movies that a user has recently watched, based on the title of the movie.

Key Steps and Functionality

1. **Data Retrieval and Preprocessing**: The code begins by importing essential libraries and loading the preprocessed movie datasets and similarity matrices.

2. **User Interaction**: The user is presented with a selection box where they can choose a movie title.

3. **Recommendation Function**: Upon clicking the "Recommend" button, the "Hybrid-recommender" function is called with the selected movie title.

4. **Movie Similarity and Filtering**: The function identifies the movie index, calculates the 100 most similar movies, and filters them based on vote counts using the weighted rating (WR) formula.

5. **Content-Based Filtering**: The function further filters the top 50 movies based on content similarity using movie overviews. It preprocesses the overviews by removing numbers and symbols and converting text to lowercase.

6. **TF-IDF Vectorization**: TF-IDF vectorization is applied to extract features from movie overviews, and cosine similarity is calculated between the movies.

7. **Recommendation Output**: The top 5 recommended movies are selected, and their posters are fetched from an external API.

8. **Streamlit Interface**: The Streamlit interface allows the user to select a movie and receive recommendations. The recommended movies and their posters are displayed in a user-friendly format.

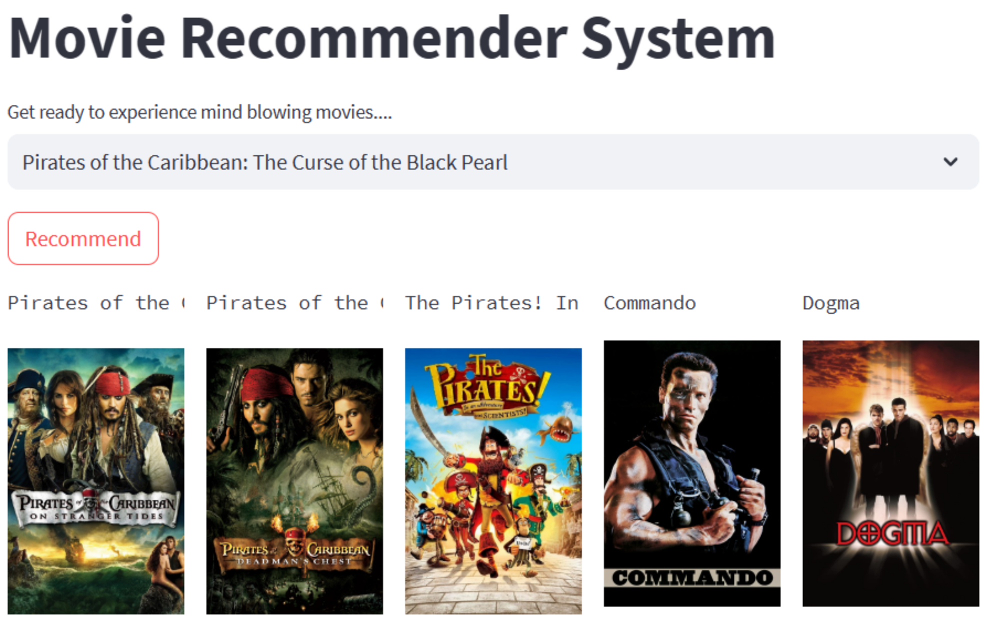

Movie Recommender System
Get ready to experience mind blowing movies....
Pirates of the Caribbean: The Curse of the Black Pearl
Recommend
Pirates of the
Pirates of the
The Pirates! In
Commando
Dogma

Figure 3.1: Hybrid Movie Recommender System

This implementation provides a user-friendly interface for our hybrid movie recommender system, offering users the ability to discover and enjoy movies that align with their preferences. The system leverages both collaborative and content-based filtering, ensuring that the recommendations are not only personalized but also of high quality and thematic relevance. The recommendations are presented in a visually appealing manner, enhancing the overall user experience.

Chapter 4

Summary and Conclusion

The project has successfully addressed the significant challenges outlined in the problem statement by developing a robust hybrid movie recommender system. This system seamlessly combines content-based and collaborative filtering methods to resolve the following key issues:

1. **Information Overload**:Our hybrid recommender system alleviates the burden of information overload by providing users with tailored movie recommendations. It enables users to navigate the extensive library of movies and discover those that truly match their preferences.

2. **Diverse User Preferences**:Recognizing the diverse and evolving nature of user preferences, our system embraces the dynamic nature of individual interests. By blending multiple recommendation techniques, it adapts to users' changing tastes.

3. **Cold Start Problem**:Our solution goes beyond traditional collaborative filtering to address the "cold start" problem effectively. New users, with limited interaction history, receive meaningful recommendations based on content-based attributes.

4. **Limited Exploration**: The hybrid approach ensures that users are not confined to a narrow range of recommendations. By combining content-based and collaborative filtering, our system offers diverse movie suggestions, enhancing user engagement and satisfaction.

The system's interface, built using Streamlit, offers an intuitive user experience, allowing users to select a movie title and instantly receive top 5 movie recommendations based on their recent viewing choices. These recommendations are not only personalized but also encompass thematic and content-related aspects, ensuring user satisfaction.

In conclusion, our hybrid movie recommender system represents an effective solution for enhancing user engagement and satisfaction in the world of digital entertainment. By integrating both collaborative and content-based filtering, we strike a balance between serendipity and relevance in movie recommendations.

This project underscores the significance of harnessing user interactions, content features, and advanced algorithms to deliver tailored and high-quality movie suggestions. It also highlights the importance of data preprocessing, feature extraction, and similarity calculations to achieve accurate and meaningful recommendations.

Our hybrid recommender system represents an effective solution for enhancing user engagement and satisfaction, reducing information overload, and accommodating diverse user preferences. As users explore the extensive library of movies, our system empowers them to discover and enjoy movies that resonate with their unique tastes and preferences. In a world of abundant content choices, our project contributes significantly to the user experience, ensuring that users receive high-quality and personalized movie recommendations.

The project's holistic approach, from data exploration to system implementation, positions it as a valuable tool for streaming platforms, cinema websites, and movie enthusiasts. It maximizes user satisfaction, retention, and exploration, making it a prominent player in the realm of digital entertainment.

Bibliography

1. Chatgpt (for drafting purpose only).
2. github.com/spoluan/TMDB-5000-Movie-recommendation-system
3. kaggle/datasets/tmdb/tmdb-movie-metadata